kassel
university
press

Entwicklung einer Software-Architektur für Systeme zum integrierten simulationsbasierten Assessment des globalen Wandels

Marcel Boris Endejan

Die vorliegende Arbeit wurde vom Fachbereich Elektrotechnik - der Universität Kassel als Dissertation zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.) angenommen.

Erster Gutachter: Prof. Dr. Joseph Alcamo
Zweiter Gutachter: Prof. Dr. Heinrich Werner
3. Prof. Dr. Werner Kleinkauf
4. Prof. Dr. Andreas Ernst

Tag der mündlichen Prüfung 22. September 2003

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar

Zugl.: Kassel, Univ., Diss. 2003
ISBN 3-89958-046-X

www.upress.uni-kassel.de

Umschlaggestaltung: M. B. Endejan, unter Verwendung einer Zeichnung von Wolfram Gothe, Hamburg (Wolfram.Gothe@architektur-zeichnung.de) / 5 Büro für Gestaltung, Kassel
Druck und Verarbeitung: Unidruckerei der Universität Kassel
Printed in Germany

Danksagung

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am wissenschaftlichen Zentrum für Umweltsystemforschung der Universität Kassel. Dem Direktor des Zentrums, Herrn Prof. Dr. Joseph Alcamo, danke ich für seine Unterstützung während der Anfertigung der Dissertation und für die Übernahme des Gutachtens. Für die freundliche Bereitschaft zur Anfertigung des Zweitgutachtens danke ich Herrn Prof. Dr. Heinrich Werner. Für die Bereitschaft zur Mitwirkung in der Promotionskommission danke ich Herrn Prof. Dr. Werner Kleinkauf und Herrn Prof. Dr. Andreas Ernst.

Meinen Kolleginnen und Kollegen am Wissenschaftlichen Zentrum für Umweltsystemforschung danke ich für die gute Arbeitsatmosphäre, die stete Diskussionsbereitschaft und die Möglichkeit, einige in der Arbeit vorgestellte Konzepte in der Praxis erproben zu dürfen. Herrn Rüdiger Schaldach danke ich für die Diskussionen sowie für die Durchsicht des Manuskripts und seine kritischen Anmerkungen. Für die Unterstützung bei der Realisierung einiger Programme danke ich Herrn Stephan Lauer. Herrn Achim Manche sei gedankt für die vielen Hilfestellungen bei technischen Fragen und Problemen. Ein ganz besonderer Dank geht an Frau Dr. Martina Flörke und Herrn Dr. Michael Märker, die mit ihren Anmerkungen und Vorschlägen und ihrem unermüdlichen Interesse einen entscheidenden Beitrag zur Gestalt der vorliegenden Arbeit geleistet haben.

Meinen Mitbewohnerinnen Vesna Jokić, Meike Siebert und Helena Siebert sowie meinem Mitbewohner Ljubomir Adžić danke ich für die sehr schöne Atmosphäre während der letzten zwei Jahre. Herrn Ljubomir Adžić danke ich auch dafür, dass er mir bei einigen Realisierungsfragen mit Rat und Tat zur Seite stand.

Der Weg zur Anfertigung dieser Arbeit war nicht immer klar. Daher möchte ich mich bei all jenen bedanken, die mich – bewusst oder unbewusst – auf diesen Weg gebracht haben und mich ermutigten, ihn weiterzugehen. So danke ich meinen Eltern Frau Karin Ritter und Herrn Heinz Josef Endejan für ihre Unterstützung und für die Möglichkeit der freien Wegwahl. Mein innigster Dank geht an Frau Klara Raas und Herrn Jakob Raas, die mich von den ersten Schritten an begleitet und ihnen Halt gegeben haben. Sehr herzlich danken möchte ich meiner Freundin Dagmar Ritterbusch, die selbst in der Abschlussphase der Arbeit viel Geduld und Verständnis gezeigt hat und – trotz der räumlichen Entfernung – immer unterstützend an meiner Seite war.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Hintergrund . . . 1

1.2 Rahmen und Ziel der Arbeit . . . 3

1.3 Struktur der Arbeit . . . 4

2 Grundlagen **7**

2.1 Integriertes Assessment . . . 7

2.1.1 Globaler Wandel . . . 7

2.1.2 Assessment des globalen Wandels . . . 11

2.2 Modellierung . . . 14

2.3 Software-Entwicklung . . . 16

2.3.1 Entwicklungsphasen . . . 16

2.3.2 Software-Architektur . . . 18

3 Stand der Technik **25**

3.1 Integrierte Modelle . . . 25

3.1.1 Definitionen . . . 25

3.1.2 Systeme . . . 28

3.1.3 Frameworks . . . 31

3.1.4 Entwicklungsumgebungen . . . 38

3.2 Standards . . . 41

3.2.1 Standardisierungs-Organisationen . . . 41

3.2.2 High Level Architecture (HLA) . . . 47

3.2.3 NIST/ECMA-Referenz-Modell . . . 52

3.2.4 Open Distributed Processing – Reference Model . . . 52

3.2.5 OpenGIS Service Architecture . . . 55

3.3 Fazit . . . 63

4 Systemdefinition **69**

4.1 OOA-Modell . . . 69

4.1.1 Gesamtmodell . . . 70

4.1.2 SISA-Ressourcen . . . 72

4.2 Anforderungsdefinition . . . 75
4.2.1 Allgemeine Anforderungen . . . 75
4.2.2 Ziele und Funktionen . . . 78
4.2.3 System-Einsatz . . . 80
4.2.4 System-Umgebung . . . 82
4.2.5 System-Daten . . . 84
4.2.6 System-Leistungen . . . 88
4.2.7 Benutzungsschnittstellen . . . 89
4.2.8 Qualitäts-Zielbestimmung . . . 89
4.2.9 Testszenarien . . . 93
4.2.10 Entwicklungs-Umgebung . . . 93
4.3 Fazit . . . 94

5 Architektur-Entwicklung 95
5.1 Komponenten-Übersicht . . . 96
5.2 Komponenten-Entwicklung . . . 96
5.2.1 Katalogmanager . . . 96
5.2.2 Metadaten-Sammler . . . 116
5.2.3 Dokumentation . . . 119
5.2.4 Simulationslaufmanager . . . 128
5.2.5 Simulationssystem . . . 131
5.2.6 Datenzugriff und Datenbanksystem . . . 134
5.2.7 Geodatenverarbeitung . . . 138
5.2.8 Datenverarbeitung . . . 141
5.2.9 Aufgabensteuerung . . . 142
5.2.10 Ergebnisanalyse . . . 144
5.2.11 Modellanalyse . . . 145
5.3 Gesamtarchitektur . . . 148
5.3.1 Komponenten . . . 149
5.3.2 Interaktionen . . . 154
5.4 Fazit . . . 160

6 Realisierung 165
6.1 Beispielmodell GLASS . . . 165
6.2 Komponenten-Übersicht . . . 168
6.3 Komponenten-Realisierung . . . 169
6.3.1 Dokumentation . . . 169
6.3.2 Katalogmanager . . . 173
6.3.3 Simulationssystem . . . 183
6.3.4 Simulationslaufmanager . . . 187
6.3.5 Datenzugriff und Datenbasis . . . 190
6.3.6 Datenverarbeitung . . . 198
6.4 Fazit . . . 199

7 Zusammenfassung und Ausblick **205**
7.1 Zusammenfassung 205
7.2 Ausblick 211

Literaturverzeichnis **213**

A Glossar **229**

B Datenmodelle und Schnittstellen **233**
B.1 Datenmodell zu Personen und Organisationen 234
B.2 Zusammenfassung des SISA-Datenmodells 235
B.3 Zusammenfassung der SISA-Schnittstellen 236

C Standards **237**
C.1 Übersichten 237
C.2 HLA-Regeln 243

D Programm-Quelltexte **245**
D.1 PHP-Beispiel 245
D.2 Metadaten-Datei 247
D.3 Metadaten-Sammler 249
D.4 Simulationseinstellungen und Datenzugriff 250
D.5 Simulationssystem 254
D.6 Datenzugriff 257
D.7 Geodatenverarbeitung 257

Abbildungsverzeichnis

2.1 Grundstruktur des globalen Beziehungsgeflechts 9
2.2 Erstellung eines Simulationsmodells 15

3.1 Architektur des Systems *GLOBESIGHT* 30
3.2 Architektur Object Modeling System 32
3.3 Schematische Modellstruktur von *PRISM* 34
3.4 Systemarchitektur von *DANUBIA* 37
3.5 Architektur der Modellumgebung *M* 39
3.6 High Level Architekture – Übersicht 50
3.7 HLA-Prinzip der Daten- und Interaktionsweitergabe 51
3.8 NIST/ECMA Referenz-Architektur 53
3.9 Statisches Modell des Dienste-Konzepts der ISO/DIS 19119 . . 57
3.10 Logische Vier-Schichten-Architektur der ISO/DIS 19119 62
3.11 Physikalische Mehrschichten-Architektur der ISO/DIS 19119 . 63

4.1 OOA-Modell eines Systems zum simulationsbasierten integrierten Assessments (SISA) 73
4.2 Ressourcen eines SISA 76
4.3 Anwendungsfalldiagramm des SISA 83
4.4 Anforderung an die System-Daten 85

5.1 Übersicht der SISA-Komponenten 97
5.2 Verbindung zwischen SISA-Komponenten und SISA-Zielen . . . 99
5.3 Georäumliche Ressourcen im Sinne des OpenGIS-Konsortiums 101
5.4 Von den *OpenGIS Catalog Services* genutzte Klassen 102
5.5 Schnittstellen-Software zum Einsatz von *OpenGIS Services* . . 104
5.6 Schnittstellen des Katalogmanagers 107
5.7 Datenmodell des Katalogs und der Katalogeinträge 108
5.8 Minimaler Elementsatz der ISO/DIS 19115 112
5.9 Datenmodell zur Speicherung von Metadaten 117
5.10 Prinzip des Metadaten-Sammlers 118
5.11 Schnittstelle des Metadaten-Sammlers 119

5.12 Datenmodell des Metadaten-Sammlers . . . 120
5.13 Schnittstellen der Dokumentationskomponente . . . 121
5.14 Datenmodell für Personen/Organisationen . . . 123
5.15 Datenmodell für Projekte u. Simulationsstudien . . . 124
5.16 Datenmodell für Szenario-Informationen . . . 126
5.17 Datenmodell für Simulationsläufen . . . 126
5.18 Datenmodell für Aufgaben-Informationen . . . 127
5.19 Datenmodell für Anmerkungen . . . 128
5.20 Simulationsmodelle und deren Einstellungen . . . 129
5.21 Schnittstellen des Simulationslaufmanagers . . . 130
5.22 Zusammenhang Simulationsmodell u. Modelleinstellungen . . . 131
5.23 Datenmodell des Simulationslaufmanagers . . . 132
5.24 Schnittstellen des Simulationssystems . . . 134
5.25 Komponenten zum Datenzugriff und zur Datenhaltung . . . 136
5.26 Schnittstelle des Datenbanksystems . . . 137
5.27 Schnittstelle der Datenzugriffskomponente . . . 138
5.28 Datenmodell der Datenzugriffskomponente . . . 138
5.29 Schnittstellen der Komponente zur Geodatenverarbeitung . . . 141
5.30 Schnittstelle zur allgemeinen Datenverarbeitung . . . 142
5.31 Schnittstelle zur Aufgabensteuerung . . . 144
5.32 Schnittstelle der Analyse-Komponente . . . 145
5.33 Prinzip der Modellanalyse . . . 147
5.34 Schnittstellen zur Modellanalyse . . . 148
5.35 Einfaches Datenmodell zur Modellanalyse . . . 148
5.36 Komponenten der Architektur . . . 151
5.37 Architektur-Dynamik – Erzeugung von Simulationsergebnissen 155
5.38 Architektur-Dynamik – Sammlung von Metadaten . . . 157
5.39 Architektur-Dynamik – Analyse eines Modells . . . 159

6.1 Struktur des GLASS-Modells . . . 167
6.2 Tabelle zur Speicherung der Anmerkungen . . . 170
6.3 Übersicht zum Realisierungsbeispiel der SISA-Komponenten . . 171
6.4 Datenbankzugriff über Web-Browser . . . 172
6.5 Dokumentation der Daten über Personen/Organisationen . . . 173
6.6 Anzeige der Daten über Personen/Organisationen . . . 174
6.7 Uniform Resource Identifiers . . . 175
6.8 Web-Seite zur Generierung eindeutiger Namen . . . 178
6.9 Anzeige der Ressourcen-Liste . . . 180
6.10 Web-Seite zur Metadaten-Erfassung . . . 181
6.11 Web-Seite zur Anzeige von Metadaten . . . 184
6.12 Basisklasse der Simulationsmodelle . . . 185
6.13 Prinzip eines Adapters . . . 186
6.14 Kommunikation über Sockets . . . 188

6.15 Realisierung des Simulationslaufmanagers 190
6.16 Prinzip der Mediatoren 192
6.17 Funktionsbeispiel zur Geodatenverarbeitung 199

B.1 Datenmodell zu Personen und Organisationen 234
B.2 Übersicht des SISA-Datenmodells 235
B.3 Übersicht der SISA-Koponenten und Schnittstellen 236

C.1 OpenGIS Abstract Spezifikations – Abhängigkeiten 237

Tabellenverzeichnis

2.1 Qualitätsmerkmale und Qualitäts-Teilmerkmale der ISO/IEC 9126 23

3.1 Sichtweise auf ein Software-System nach ISO/RM-ODP 54

4.1 Arbeitsschritte einer Simulationsstudie 72
4.2 Grundlegende SISA-Ziele 81
4.3 Grundlegende SISA-Funktionen 81
4.4 Grundlegende SISA-Datenbestände 88
4.5 Anforderungen des SISA an die Software-Qualität. 90
4.6 Vorrangige nicht-funktionale Anforderungen an das SISA . . . 93

5.1 Komponenten und ihre Verantwortlichkeiten 98
5.2 Die primären Funktionen der *OpenGIS Catalog Services* 103
5.3 Schnittstellen und Operationen der Katalog-Komponente . . . 106
5.4 Pakete des Metadaten-Standards ISO/DIS 19115 109
5.5 Kern-Metadaten der ISO/DIS 19115 110
5.6 Übersicht zum *Content Standard for Computational Models* . . 114
5.7 Metadaten-Elementsatz der ISO 15836 (Dublin Core) 116

6.1 ISO-Standards zur Datenkodierung 198

C.1 Beipieldienste der ISO/DIS 19119 Taxonomie, Teil I 238
C.2 Beipieldienste der ISO/DIS 19119 Taxonomie, Teil II 239
C.3 Arbeitsprogramme des ISO/TC 211 240
C.4 OpenGIS Abstract Spezifications 241
C.5 OpenGIS Implementation Spezifications 242

Abkürzungsverzeichnis

ADEPT Alexandria Digital Earth Prototype

ADO Active Data Objects

API Application Programming Interface

ASCII American Standard Code for Information Interchange

CSCM Content Standard for Computational Models

DCMES Dublin Core Metadata Element Set

DBMS Datenbank-Managementsystem

DIN Deutsches Institut für Normung

DOM Document Object Model

FGDC Federal Geographic Data Committee

GIS Geo-Informationssystem

GLASS Global Assessment of Security

GML Geographical Markup Language

GUI Graphical User Interface

HLA High Level Architecture

HYDE Hundred Year Database for Integrated Environmental Assessments

HTML Hyper-Text Markup Language

HTTP Hyper-Text Transport Protocol

IEEE Institute of Electrical and Electronics Engineers

IMAGE Integrated Model to Assess the Greenhouse Effect

IPCC Intergovernmental Panel on Climate Change

ISO International Organization for Standardization

ISO/DIS ISO Draft International Standard

ISO/TC ISO Technical Committee

ODBC Open Database Connectivity

ODP Open Distributed Processing

OGC Open GIS Consortium

OMS Object Modeling System

OOA Objekt-orientierte Analyse

PHP PHP: Hypertext Preprocessor (rekursive Abkürzung)

RDBMS Relationales Datenbank-Managementsystem

RDF Resource Description Framework

RIVM Rijksinstituut voor Volksgezondheid en Milieu (staatliches Institut für Gesundheit und Umwelt der Niederlande)

RM-ODP Open Distributed Processing – Reference Model

SI International System of Units

SISA System zum integrierten simulationsbasierten Assessment

SOF Service Organizer Folder

UML Unified Modeling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

URN Uniform Resource Name

W3C World Wide Web Consortium

WWW World Wide Web

XML Extensible Markup Language

Kapitel 1

Einleitung

1.1 Hintergrund

Die Zunahme der Weltbevölkerung, die langfristig veränderte Zusammensetzung der Atmosphäre, der Rückgang der biologischen Vielfalt sowie die Degeneration und der Verlust von Böden sind nur Beispiele der Veränderungen, die unter dem Begriff des 'globalen Wandels' zusammengefasst werden. Die mit dem globalen Wandel einhergehenden Veränderungen und die daraus resultierenden Probleme bedürfen zu ihrer Analyse und Lösung detaillierter und in sich konsistenter Bewertungen (Luiten, 1999). Die Rolle der Informationstechnik bei der Unterstützung umweltpolitischer Entscheidungen wurde bereits bei der ersten Konferenz der Vereinten Nationen für Umwelt und Entwicklung in *Rio de Janeiro 1992* erkannt. Im Kapitel 40 (Informationen für die Entscheidungsfindung) des aus der Rio-Konferenz hervorgegangenen Aktionsprogramms *Agenda 21* (UN, 1992) wird die 'Überbrückung der Datenlücke' und die 'Verbesserung der Verfügbarkeit von Informationen' gefordert. Dies soll u. a. durch die Nutzung geographischer Informationssysteme (GIS) und die Verwendung von Expertensystemen und Modellen erreicht werden. globaler Wandel

Der Einsatz von GIS und Modellen hat sich bereits in unterschiedlichen Disziplinen und Problembereichen (Sektoren) etabliert – beispielsweise in der Ökologie sowie den Forst- und Agrarwissenschaften. Darüber hinaus werden seit Mitte der 1980er Jahre auch Modelle entwickelt, die eine sektorübergreifende Problemanalyse erlauben. Diese so genannten *integrierten Modelle* stellen eine *interdisziplinäre* Abbildung von *Ursache-Wirkungs-Ketten* dar (vgl. Parson, 1995; van der Sluijs, 1996). Integrierte Modelle bieten aufgrund der Integration der wichtigsten, für eine Fragestellung relevanten Disziplinen einen Mehrwert gegenüber disziplinären Studien und sollen lösungsrelevante Informationen für Entscheidungsträger bereitstellen. Die Modelle RAINS (Alcamo u. a., 1990) und Integrierte Modelle

Besonderheiten

IMAGE2 (Alcamo, 1994) stellen zwei wichtige Vertreter solcher Simulationsmodelle dar (vgl. Weyant u. a., 1996). Integrierte Modelle unterscheiden sich von anderen Simulationsmodellen nicht nur durch die Berücksichtigung mehrerer *Fachdisziplinen*, sondern auch durch den großen *räumlichen Maßstab* der abgebildeten Prozesse, der sich üblicherweise zwischen dem nationalen und globalen befindet, sowie durch den simulierten *Zeitraum*, der sich von Dekaden bis hin zu mehr als einem Jahrhundert erstrecken kann (vgl. Parson, 1995; Weyant u. a., 1996; Bakkes u. a., 2000).

Komplexität

Aufgrund der sich stetig verbessernden digitalen *Datengrundlage* und *Rechnerleistungen* und dem vermehrten Aufkommen quantitativer *Systembeschreibungen* einzelner Komponenten des *Systems Erde*, werden die integrierten Modelle – umfassender ausgedrückt: Systeme zum simulationsbasierten integrierten Assessment – immer *detailgetreuer*, *komplexer* und *leistungsfähiger* und gewinnen zunehmend an Bedeutung. Schneider (1997) sieht die integrierten Modelle gar als die primären Werkzeuge für Analysen im Rahmen des globalen Wandels.

Anforderungen

Durch die erhöhte Komplexität und Größe von integrierten Modellen ergeben sich Herausforderungen bezüglich der *Transparenz*, der *Nachvollziehbarkeit* und der *Reproduzierbarkeit* von Analysen und Ergebnissen sowie der *Erweiterbarkeit* von Modellen und der *Wiederverwendbarkeit* und *Austauschbarkeit* von Modellteilen. Eine weitere Anforderung – die der *Interoperabilität* – ergibt sich aus der notwendigen Zusammenarbeit mit anderen Software-Systemen (wie beispielsweise GIS), die zur Vorverarbeitung oder Nachbearbeitung von Daten benötigt werden. All diese Aspekte bestimmen die *Qualität* eines integrierten Modells und müssen daher bei der Entwicklung eines solchen Software-Systems berücksichtigt werden.

Problem

Trotz der gestiegenen Anforderungen sind vorhandene Modelle zumeist unzureichend dokumentiert und modularisiert (Jaeger u. a., 2002) und folgen i. d. R. keinen anderenorts bereits etablierten Standards. Die *Austauschbarkeit*, *Wartbarkeit* und *Wiederverwendbarkeit* von Modellteilen und Daten ist daher sehr gering und der *Arbeits- und Kostenaufwand* zur Erstellung neuer Modelle sehr hoch. Benz u. a. (1997) führen das Fehlen einwandfreier Modelldokumentationen darauf zurück, dass deren Erstellung sehr zeitaufwendig ist und dass Wissenschaftler für eine Dokumentation weder 'belohnt' noch für eine fehlende 'bestraft' werden. Jaeger u. a. (2002) sehen als Grund für die mangelhafte Dokumentation und Transparenz den Umstand, dass Modellentwickler in erster Linie Wissenschaftler und keine Software-Ingenieure sind.

Aufgrund der Hürden, die zur Wiederverwendung vorhandener Modellteile überwunden werden müssen, wird es bei neuen Projekten oft vorgezogen, die Modelle von Grund auf neu zu entwickeln (Rizzoli und Davis, 1999). Dieser Weg der Software-Entwicklung ist weder zeitgemäß noch genügt er den gestiegenen

Anforderungen integrierter Modelle und bedarf daher einer richtungsweisenden Korrektur.

1.2 Rahmen und Ziel der Arbeit

Konzept

Um die Komplexität und die damit einhergehenden Probleme integrierter Modelle in den Griff zu bekommen, werden Prinzipien und Methoden der *Software-Technik* angewendet. Da die Qualität großer Software-Systeme mit deren *Architektur* steht und fällt (Foegen und Battenfeld, 2001), ist das grundlegend anzuwendende Prinzip das der *Modularisierung* – ein integriertes Modell muss also in seine grundlegenden Module (Komponenten) zerlegt werden. Die grundlegenden Komponenten ergeben, zusammen mit einer Definition der Verantwortlichkeiten der einzelnen Komponenten im Gesamtsystem und einer Spezifikation der Schnittstellen, über die die Komponenten angesprochen werden, die *Software-Architektur*.

Ziel

Ziel der Arbeit ist eine allgemein anwendbare *Software-Architektur* für Systeme zum integrierten simulationsbasierten Assessment, die die *Wiederbenutzbarkeit* und *Wiederbenutzung* von Modellen, Modellteilen, Daten und anderen notwendigen *Betriebsmitteln* unterstützt, die *Zusammenarbeit* mit anderen Programmen begünstigt und die *Qualität* der Ergebnisse sichern hilft.

Fragen

Um dieses Ziel zu erreichen, stellen sich – unter der Annahme, dass sich integrierte Modelle in einzelne Komponenten aufteilen lassen – die folgenden Forschungsfragen:

- In welche generellen Komponenten sollte ein System zum integrierten simulationsbasierten Assessment aufgeteilt werden?
- Welche Komponenten können unabhängig von *einem* konkreten System *realisiert* und damit für unterschiedliche Modelle wieder verwendet werden?
- Welche Daten sollten zur Unterstützung der Transparenz von Analyse- und Simulationsergebnissen vorgehalten werden?
- Welche Standards können zur Erhöhung der Qualität integrierter Modelle beitragen?

Methode

Da es keine etablierten Architekturen für integrierte Modelle gibt (vgl. Jaeger u. a., 2002), bilden veröffentlichte *Modellübersichten* und detaillierte *Modellbeschreibungen* den Rahmen für die Komponentenbildung und die Beantwortung der Forschungsfragen. Mit Hilfe der Modellbeschreibungen werden Funktionalitäten und Strukturen integrierter Modelle identifiziert, die als existentiell für jedes integrierte Modell angesehen werden können. Basierend auf diesen Informationen erfolgt eine 'Systemdefinition', die die grundlegenden Anforderungen und Ziele eines integrierten Modells formal definiert. Für dieses 'allgemeine' integrierte Modell wird dann eine Software-Architektur entwickelt.

Die Entwicklung der Architektur erfolgt, aus Gründen der Austauschbarkeit und Interoperabilität, unter Berücksichtigung relevanter *Standards*. Die Überprüfung der Architektur erfolgt durch eine prototypische Implementierung der Komponenten im Rahmen der Entwicklung eines ‘konkreten’ integrierten Modells.

1.3 Struktur der Arbeit

Grundlagen

Das folgenden Kapitel der *Grundlagen* beginnt mit einigen Ausführungen zum thematischen Hintergrund der hier betrachteten Software-Systeme und definiert die zentralen Begriffe *‘globaler Wandel’* und *‘integriertes Assessment’*. Zur Entwicklung von *Software-Systemen* gibt es *Phasenmodelle*, die den Prozess zum fertigen Produkt hin beschreiben – die Software-Architektur ist das Ergebnis einer dieser Phasen (der Entwicklungsphase). Um die einzelnen Schritte der vorliegenden Arbeit in diesen Prozess einordnen zu können, wird ein solches Phasenmodell vorgestellt. Anschließend folgen weitere Erklärungen zum Begriff der *Software-Architektur*. Für die Erstellung von *Simulationsmodellen* gibt es ebenfalls Phasenmodelle. Anhand eines dieser Phasenmodelle werden einige grundlegende Begriffe der Modellierung erklärt.

Stand der Technik

Kapitel 3 (Seite 25) gibt einen Überblick über den *Stand der Technik* im Bereich der integrierten Modellierung und über Arbeiten, die im Zusammenhang mit der Erstellung der Software-Architektur relevant sind. Es werden einige *integrierte Modelle* und *Hilfsmittel* (Frameworks und Entwicklungsumgebungen) vorgestellt, die im Rahmen der integrierten Modellierung entwickelt wurden. Darüber hinaus werden einige *Standards* angesprochen, deren Berücksichtigung die Interoperabilität integrierter Modelle steigern kann.

Systemdefinition

Die Basis für die Entwicklung eines Software-Systems ist die *Definition der Ziele und Leistungen* sowie der gewünschten *Qualitätsmerkmale* des Systems. Die Systemdefinition in Kapitel 4 (Seite 69) beginnt mit einer objektorientierte Analyse eines integrierten Modells und dessen Systemumgebung. Das daraus resultierende *Objekt-Modell* dient dem besseren Verständnis der Gesamtzusammenhänge eines simulationsbasierten Assessments und der Einordnung und Definition wichtiger Begrifflichkeiten. Anschließend folgt eine *Anforderungsdefinition*, in der u. a. die grundlegenden Ziele und Funktionen eines Systems zum integrierten simulationsbasierten Assessment aufgeführt werden.

Architektur

Aufbauend auf der Systemdefinition wird in Kapitel 5 (Seite 95) eine *Software-Architektur* für das spezifizierte System entwickelt. Das Gesamtsystem wird hier in einzelne Komponenten gegliedert, wobei die einzelnen *Komponenten* eine definierte *Zuständigkeit* innerhalb des Gesamtsystems erhalten. Neben der Festlegung der Verantwortlichkeit der Komponenten werden wichtige *Datenstrukturen* der Komponenten definiert sowie die *Schnittstellen*, über die sie miteinander verbunden sind. Die sich aus den einzelnen Komponenten und de-

ren Zusammenspiel ergebenden statischen und dynamischen Zusammenhänge der Architektur werden abschließend zusammengefasst und liefern damit eine Gesamtübersicht über die entwickelte Architektur.

Realisierung

Eine prototypische *Realisierung* zentraler Komponenten der entwickelten Architektur folgt in Kapitel 6 (Seite 165). Das Kapitel beginnt mit einer kurzen Beschreibung des integrierten Modells *GLASS*, anhand dessen die Implementierung der zentralen Teile der Architektur beschrieben wird. Anschließend wird die Realisierung der Architektur-Komponenten im Einzelnen beschrieben. Das Fazit dieses Kapitels fasst die wichtigsten Punkte der Komponenten-Realisierung zusammen und stellt die in der Systemdefinition aufgestellten Anforderungen den erzielten Resultaten gegenüber.

Zusammenfassung

Das letzte Kapitel (Kapitel 7, Seite 205) liefert eine *Zusammenfassung* der wichtigsten Erkenntnisse dieser Arbeit und zeigt Möglichkeiten der Erweiterung und Verfeinerung der Architektur auf.

Kapitel 2

Grundlagen

Dieses Kapitel beschreibt einige Grundlagen, die zum Verständnis des Problembereichs und der Entwicklung der Software-Architektur hilfreich sind.

Systeme zum simulationsbasierten integrierten Assessment werden zur Analyse von Problemstellungen des globalen Wandels eingesetzt. Das Kapitel beginnt daher mit einer Definition und Abgrenzung dessen, was als *globaler Wandel* verstanden wird. Anschließend wird die Bedeutung des Begriffs *Assessment* genauer beleuchtet, der im Zusammenhang mit Studien zum globalen Wandel eine besondere Bedeutung besitzt. Da die Arbeitsgrundlage für ein simulationsbasiertes integriertes Assessment die Ergebnisse von *Simulationsmodellen* sind, werden in Abschnitt 2.2 (Seite 14), anhand einer Vorgehensweise zur Erstellung von Simulationsstudien, wichtige *Grundbegriffe der Modellierung* erklärt. Die zum besseren Verständnis der weiteren Ausführungen notwendigen Grundlagen der *Software-Entwicklung* sind Thema des letzten Abschnitts dieses Kapitels (Abschnitt 2.3, Seite 16). Hier werden u. a. die einzelnen *Phasen* beschrieben, die bei der Entwicklung eines Software-Systems zu durchlaufen sind. Die Aspekte, die bei der Entwicklung einer *Software-Architektur* eine Rolle spielen, werden zum Abschluss des Kapitels behandelt. Übersicht

2.1 Integriertes Assessment

2.1.1 Globaler Wandel

Im Juli 1992 fand in Rio de Janeiro die erste *Konferenz der Vereinten Nationen für Umwelt und Entwicklung* statt. Aus dieser Konferenz ist ein Aktionsprogramm für das 21. Jahrhundert hervorgegangen: die so genannte *Agenda 21* (UN, 1992). In der Präambel der Agenda 21 heißt es: Agenda 21

> Die Menschheit steht an einem entscheidenden Punkt in ihrer Geschichte. Wir erleben eine zunehmende Ungleichheit zwischen Völkern und innerhalb von Völkern, eine immer größere Armut, immer mehr Hunger, Krankheit und Analphabetentum sowie eine fortschreitende Schädigung der Ökosysteme, von denen unser Wohlergehen abhängt. Durch eine Vereinigung von Umwelt- und Entwicklungsinteressen und ihre stärkere Beachtung kann es uns jedoch gelingen, die Deckung der Grundbedürfnisse, die Verbesserung des Lebensstandards aller Menschen, einen größeren Schutz und eine bessere Bewirtschaftung der Ökosysteme und eine gesicherte, gedeihlichere Zukunft zu gewährleisten.

globaler Wandel

Die angeführten zunehmenden Ungleichheiten und Probleme sind nur einige der weltweit zu beobachtenden Veränderungen, die oft unter dem Begriff des *globalen Wandels* zusammengefasst werden.

Das Bundesministerium für Bildung und Forschung definiert den Begriff des 'globalen Wandels' recht eingängig als die „... Veränderungen in Natur und Gesellschaft, die die Menschheit als Ganzes und auf längere Sicht hin betreffen" (Krück u. a., 2001).

Um das Ziel der 'Vereinigung von Umwelt- und Entwicklungsinteressen' zu erreichen, müssen die vielfältigen Ursachen und Wirkungen, die mit den Veränderungen verknüpft sind, berücksichtigt werden. Als Ausgangspunkt für systematische Untersuchungen der Veränderungen ist eine ausführliche Definition dessen, was unter dem globalen Wandel zu verstehen ist, notwendig.

WBGU

Der im Vorfeld der Konferenz von Rio de Janeiro von der Deutschen Bundesregierung berufene *Wissenschaftliche Beirat Globale Umweltveränderungen (WBGU)* definiert in seinem ersten Jahresgutachten (WBGU, 1993) den Begriff des *globalen Wandels* wie folgt:

Def. globaler Wandel

> Der Beirat versteht unter globalen Veränderungen der Umwelt solche, die den Charakter des Systems Erde zum Teil irreversibel modifizieren und deshalb direkt oder indirekt die natürlichen Lebensgrundlagen für einen Großteil der Menschheit spürbar beeinflussen. Globale Veränderungen der Umwelt können sowohl natürliche als auch anthropogene Ursachen haben. Um diesen Gesamtzusammenhang zu kennzeichnen, wird der Begriff des globalen Wandels verwendet.
>
> Umwelt selbst wird dabei definiert als die Gesamtheit aller Prozesse und Räume, in denen sich die Wechselwirkung zwischen Natur und Zivilisation abspielt. Somit schließt 'Umwelt' alle natürlichen Faktoren ein, welche von Menschen beeinflusst werden oder diese beeinflussen.

Das *System Erde* wird vom WBGU als Kombination aller Komponenten der *Natursphäre* und *Anthroposphäre* verstanden. Abbildung 2.1 gibt einen Überblick über die einzelnen Komponenten dieser beiden Sphären, wie sie der WBGU sieht. System Erde

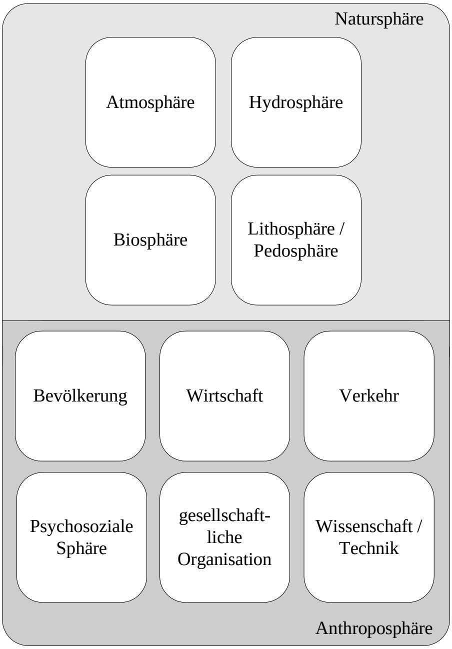


Abbildung 2.1: Grundstruktur des globalen Beziehungsgeflechts. Quelle: WBGU (1993).

Im Rahmen des globalen Wandels werden für die verschiedenen, in der Abbildung dargestellten Komponenten, u. a. die folgenden Prozesse angegeben: Prozesse

Natursphäre

Atmosphäre Reduktion stratosphärischen Ozons, verstärkter Treibhauseffekt, Troposphärenverschmutzung

Hydrosphäre Meeresspiegelanstieg, Abflussänderungen und Verlagerung von Strömungen, Senkung des Grundwasserspiegels

Biosphäre Änderung ökologischer Struktur und Leistung, Artenschwund, Reduktion von Wäldern und Feuchtgebieten, Übernutzung

Lithosphäre/Pedosphäre Erosion, Kontamination, Überdüngung, Strukturzerstörung

Anthroposphäre

Bevölkerung Bevölkerungswachstum, Altersstruktur, Urbanisierung, Migration

Wirtschaft Wirtschaftswachstum, Globalisierung der Märkte, Tertiärisierung der Produktion[1], sektoraler Strukturwandel

Verkehr Verkehrsaufkommen, Emissionsänderung, Verkehrsträgermix

Psychosoziale Sphäre Anspruchsleistung, Emanzipation, Individualisierung, wachsendes Umweltbewusstsein

Gesellschaftliche Organisation Separatismus, Fundamentalismus, Ausbreitung der Marktwirtschaft, Föderalismus, Demokratisierung

Wissenschaft/Technik Automatisierung/Mechanisierung, Effizienzsteigerung, medizinischer Fortschritt, Agrartechnologie/Biotechnologie

Neben diesen Prozessen, die direkt den einzelnen Komponenten (Hauptkompartimenten) der *Natursphäre* und *Anthroposphäre* zugeordnet werden können, werden noch Prozesse angeführt, die zwischen den Hauptkompartimenten zu finden sind. Ein Beispiel für einen solchen Prozess mit *Querschnittscharakter* ist die *Expansion/Intensivierung der Landwirtschaft*, die sowohl in enger Beziehung zum Kompartiment der Biosphäre als auch zu dem der Bevölkerung und der Wirtschaft steht.

Handlungsbedarf

Die am dringlichsten anzugehenden globalen Probleme und Haupttrends sind laut WBGU (1993): die Zunahme der Bevölkerung der Erde, die langfristig veränderte Zusammensetzung der Atmosphäre, der Rückgang der biologischen Vielfalt und die Degeneration und der Verlust von Böden. Darüber hinaus gehören Probleme durch immer größer werdende Städte und die Versorgung mit sauberem Trinkwasser – vor allem in den Entwicklungsländern – zu den wichtigsten Herausforderungen (Krück u. a., 2001).

Zur Verminderung globaler Umweltveränderungen sieht das WBGU (1993) drei Handlungsprinzipien:

- Berücksichtigung der Konsequenzen für das ganze *System Erde* bei jeder Einzelentscheidung
- Beachtung der Einheit von Umwelt und Entwicklung bei jeder politischen Entscheidung
- Ausweitung der ökonomischen Bewertungssysteme auf Naturgüter

Für eine umfassende Bewertung in diesem Sinne sind vielschichtige Auswirkungsanalysen notwendig. Verschiedene Methoden zur Erstellung solcher Analysen sind Thema des folgenden Abschnitts.

[1]Vom Schwerpunkt der Erzeugung von Nahrungsmitteln und Rohstoffen *(Primärsektor)* bzw. des Handwerks und der Industrie *(Sekundärsektor)* hin zu Dienstleistungen *(Tertiärsektor)*.

2.1.2 Assessment des globalen Wandels

Die Analyse der vielfältigen Zusammenhänge und der Auswirkungen von Änderungen innerhalb des *Systems Erde* ist aufgrund der komplexen Wirkzusammenhänge eine nicht-triviale Aufgabe. Eine Organisation, die sich mit der Erforschung der vielfältigen Ursache-Wirkungs-Ketten eines der oben angeführten Hauptprobleme, nämlich der langfristig veränderten Zusammensetzung der Atmosphäre, beschäftigt, ist das *zwischenstaatliche Gremium für Klimawandel (Intergovernmental Panel on Climate Change, IPCC)*[2]. Das IPCC wurde 1988 von der meteorologischen Weltorganisation (World Meteorological Organization, WMO) und dem Umweltprogramm der Vereinten Nationen (United Nations Environmental Programme, UNEP) gegründet. Ziel des IPCC ist die Analyse wissenschaftlicher, technischer und sozio-ökonomischer Informationen, die relevant sind zum Verständnis des durch den Menschen verursachten Klimawandels, dessen potentiellen Auswirkungen und der Optionen zur Verminderung und Anpassung (IPCC, 2001). IPCC

Das IPCC ist in drei Arbeitsgruppen unterteilt. Schwerpunkt der ersten Gruppe *(Working Group I, WG I)* ist die Beobachtung und Projizierung des Klimawandels. Die zweite Arbeitsgruppe *(WG II)* beschäftigt sich mit der Untersuchung von Vulnerabilität, Auswirkungen und Anpassungen im Zusammenhang mit dem Klimawandel. Thema der dritten Arbeitsgruppe *(WG III)* sind die möglichen Optionen zur Minderung des Klimawandels. Die *WG II* muss aufgrund ihres Arbeitsschwerpunktes in besonderem Maße das gesamte *System Erde*, also sowohl die Natursphäre als auch die Anthroposphäre und deren Verbindungen, betrachten. Zu diesem Zweck setzt die Arbeitsgruppe u. a. das so genannten *'integrierte Assessment'* ein. WG II

In der Literatur sind unterschiedliche Definitionen des Begriffs 'integriertes Assessment' (englisch: integrated assessment)[3] zu finden (s. z. B. Alcamo, 2002; Easterling, 1997; IPCC, 2001; Parson, 1995; Peirce, 1998; Rotmans, 1998; Tol und Vellinga, 1998; van der Sluijs, 1996; Weyant u. a., 1996). integriertes Assessment

Das IPCC sieht das integrierte Assessment als einen interdisziplinären Prozess, der das Ziel verfolgt, komplizierte Systeme besser zu verstehen und definiert den Begriff wie folgt (IPCC, 2001): *Definitionen*

> Integrated assessment is an interdisciplinary process that combines, interprets, and communicates knowledge from diverse scientific disciplines from the natural and social sciences to investigate and

[2]Startseite im Internet: `http://www.ipcc.ch`

[3]Für das englische Wort *assessment* gibt es kein deutsches Wort, das dem Bedeutungsumfang des englischen gerecht wird. Das Wort wird daher als *Assessment* stehen gelassen und nicht durch einen deutschen Begriff übersetzt. *Assessment* im hier betrachteten Zusammenhang kann als *Abschätzung, Einschätzung, (Be-)Wertung, Auswertung oder Begutachtung* verstanden werden.

> understand causal relationships within and between complicated systems.

Alcamo (2002) definiert den Begriff – im Zusammenhang mit der Abgrenzung von den integrierten Modellen (s.u.) – folgendermaßen:

> „Integrated assessment“ is the assembling, analysis, and communication of knowledge from different disciplines and areas of expertise to assist policymaking; it may or may not involve models, integrated or otherwise.

Merkmale

Bei van der Sluijs (1996) werden Definitionen verschiedener Autoren aufgeführt, die sich durch die folgenden Merkmale charakterisieren lassen:

- Sektorübergreifende Abbildung der gesamten Ursache-Wirkungs-Kette des Klimawandel-Problems
- Informationsbereitstellung für Entscheidungsträger
- Verwendung und Integration eines breiten Spektrums von Untersuchungsbereichen, Methoden und Analysetechniken
- Projektion zukünftiger ökonomischer Aktivität als Ausgangspunkt der Wirkungsketten
- Kombination, Interpretation und Kommunikation von Wissen unterschiedlicher wissenschaftlicher Disziplinen
- Zusammenschluss formaler Modelle oder Studien ohne Modellierungsunterstützung in einem in sich schlüssigen, konsistenten Framework
- Bearbeitung von Fragestellungen zum globalen Wandel
- Bearbeitung von Fragestellungen zu Umweltproblemen
- Berücksichtigung von Informationen über physikalische, chemische, biologische, psychologische, sozio-ökonomische und institutionelle Phänomene, inklusive relevanter Entscheidungsfindungsprozesse

Parson (1995) sieht bei Projekten zur integrierten Analyse des Klimawandels die folgenden charakteristischen Merkmale:

- Nutzung eines nationalen bis globalen räumlichen Maßstabes, wobei die Welt in letzterem Fall typischerweise in so genannte Weltregionen eingeteilt wird
- Verwendung eines zeitlichen Maßstabes zwischen einigen Dekaden und etwa einem Jahrhundert
- Nutzung von Interpolations-, Parametrisierungs- und Näherungsverfahren zur Beschreibung von Prozessen, die auf einer feineren räumlichen oder zeitlichen Auflösung ablaufen
- eine eher grobe sektorale Auflösung, d. h. beispielsweise eine vereinfachte Darstellung der unterschiedlichen Bereiche im ökonomischen Sektor

Die herausgestellten Definitionen und Merkmale setzen unterschiedliche Schwerpunkte, z. B. auf den Prozess, die Zielgruppe oder die zu betrachtenden Disziplinen. Auf einige der angeführten Merkmale wird im Kapitel der Systemdefinition (Kapitel 4, Seite 69) noch einmal eingegangen. Im Rahmen der vorliegenden Arbeit soll der Begriff des 'integrierten Assessments' zusammenfassend wie folgt verstanden werden:

> Das integrierte Assessment ist ein Prozess, in dem Wissen unterschiedlicher Fachdisziplinen über das 'System Erde' in einem konsistenten Rahmen kombiniert und interpretiert wird und der das Ziel verfolgt, den Zustand und mögliche langfristige Änderungen des Systems einzuschätzen und zu bewerten sowie die Ergebnisse politischen Entscheidungsträgern zu vermitteln. *Def.*

Methoden

Integrierte Assessments können mit verschiedenen methodischen Ansätzen durchgeführt werden. Das (IPCC, 2001) nennt hier: 1) computerunterstützte Modellierung, 2) Szenarienanalyse, 3) interaktive Computersimulation (Planungsspiele)[4], 4) teilnehmende integrierte Analyse[5] und 5) qualitative Analysen, die auf bereits existierenden Analysen und auf Expertise basieren.

Simulation

Die Möglichkeit der in sich konsistenten Abbildung der gesamten Wirkungskette eines betrachteten Problems (die so genannte 'end-to-end integration') macht integrierte Assessments so bedeutsam (Parson, 1995). Das nach Parson (1995) vorherrschende Mittel bei dieser 'end-to-end integration' ist die vom IPCC unter Punkt 1 genannte Methode der computerunterstützten Modellierung. Die sich stetig verbessernden Rechner-Leistungen, das steigende disziplinäre Verständnis einzelner Glieder der Ursache-Wirkungs-Ketten und die Verfügbarkeit sektoraler Modelle machen diese Art des Assessments immer praktikabler (Weyant u. a., 1996). Schneider (1997) sieht die hierzu verwendeten Simulationsmodelle, die so genannten *integrierten Modelle*, sogar als die primären Werkzeuge für Analysen im Rahmen des globalen Wandels an. Integrierte Modelle werden entweder durch die Kopplung existierender sektoraler Modelle erstellt oder durch die Neukonstruktion einfacherer und konsistenterer Modelle. Bei der Kopplung sektoraler Modelle bildet jedes Modell einen Teil der Ursache-Wirkungs-Kette ab, während die einfacheren Modelle die Ursache-Wirkungs-Kette von Anfang bis Ende abbilden und evtl. sogar Rückkopplungen implementieren (Parson, 1995). Es gibt aber auch komplexe integrierte Modelle, bei denen Rückkopplungen realisiert sind – z. B. das Modell IMAGE2 (s. Alcamo u. a., 1998b).

Integrierte Modelle bilden den grundlegenden Bestandteil der in dieser Arbeit betrachteten Software-Systeme. Informationen zum *Stand der Technik* und den Charakteristiken dieser Modelle finden sich in Abschnitt 3.1 ab Seite 25.

[4] Simulation gaming.

[5] Teilnehmer des 'participatory integrated assessment' sind z. B. Wissenschaftler, Politiker und andere Entscheidungsträger bzw. Interessierte (Tol und Vellinga, 1998).

2.2 Modellierung

Simulationsstudie

Das simulationsbasierte integrierte Assessment basiert auf Ergebnissen von berechenbaren Modellen (Simulationsmodellen) der Umwelt. Die Erstellung und Nutzung von Simulationsmodellen geschieht in mehreren Schritten. Steinhausen (1994) schlägt zur Durchführung von Simulationsstudien eine Vorgehensweise vor, die aus sieben Schritten besteht:

1. Problemformulierung und -analyse
2. Modellbildung
3. Datenerhebung
4. Erstellung des Computerprogramms
5. Modellvalidierung
6. Planung und Durchführung von Simulationsläufen
7. Auswertung und Implementierung der Ergebnisse

Problemanalyse

Im ersten Schritt wird das Untersuchungsproblem identifiziert und geprüft, ob die Methode der Simulation die angemessene Herangehensweise zur Lösung darstellt. Nach der Klärung dieses Punktes wird das Problem näher beschrieben und das *Simulationsziel* spezifiziert. Zur *Problembeschreibung* werden die Bestandteile des Problembereichs sowie deren Eigenschaften und Beziehungen untereinander erfasst. Außerdem werden in diesem Schritt die *Systemgrenzen* definiert, das System wird hier also von seiner Umgebung klar getrennt.

Modellbildung

Der zweite Schritt, die *Modellbildung*, beschäftigt sich mit der Reduktion und Abstraktion der Elemente des zu analysierenden Problems. In diesem Schritt werden die wichtigsten *systembeeinflussenden Elemente* ausgewählt und durch entsprechende Symbole und Regeln (Verknüpfungen zwischen den Elementen) wiedergegeben. Auf diese Weise entsteht das so genannte *konzeptionelle Modell*, das beispielsweise über ein System mathematischer Gleichungen ausgedrückt wird. Die Basiskonzepte der Software-Entwicklung (z. B. die objektorientierte Analyse) können in diesem Schritt ebenfalls sinnvoll eingesetzt werden (vgl. Unterabschnitt 2.3.1, Seite 16).

Datenerhebung

Für den Betrieb eines Simulationsmodells werden unterschiedliche Daten benötigt: 1) Daten für Modellgrößen, die während eines Simulationslaufes konstant bleiben (Systemparameter[6]), 2) Daten über systembeeinflussende exogene Größen (Modellumwelt-Größen), 3) Optionen zur Simulationssteuerung und 4) evtl. benötigte stochastische Daten (falls es sich um ein stochastisches Modell handelt, also ein Modell, das Zufallsverteilungen berücksichtigt).

Programmerstellung

Da manuelle Lösungsverfahren in den meisten Fällen zu aufwendig sind, muss das konzeptionelle Modell durch ein *Computerprogramm* realisiert werden. Die Realisierung des Modells kann über verschiedene Simulationsverfahren

[6]Bossel definiert den Begriff ‘Parameter’ etwas genauer als Größen, „... die nicht durch Veränderungen im System selbst beeinflusst sind, die oft konstant sind, aber möglicherweise auch von der Zeit abhängen.“ (Bossel, 1994).

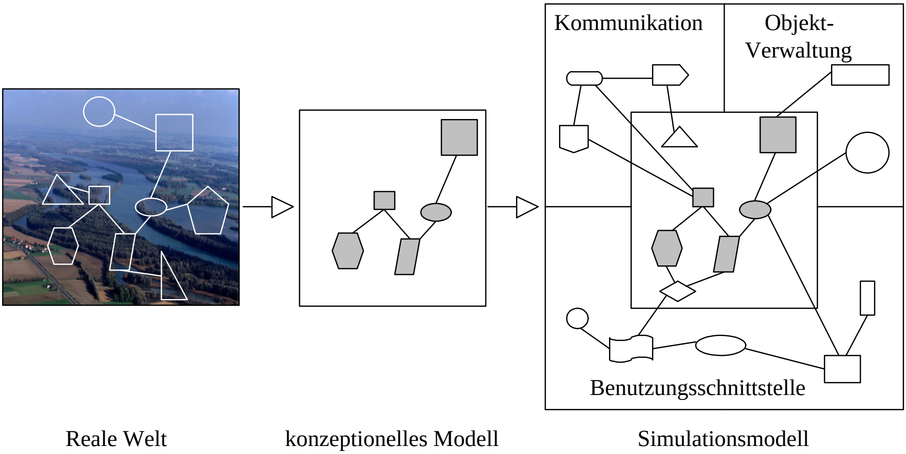


Abbildung 2.2: Erstellung eines Simulationsmodells: Von der Beschreibung eines Ausschnitts der *realen Welt* werden die wichtigsten, systembeeinflussenden Faktoren und deren Zusammenhänge herausgenommen und im *konzeptionellen Modell* beschrieben. Zur Umsetzung in ein *Simulationsmodell* sind aus softwaretechnischer Sicht Elemente hinzuzufügen um anschließend Simulationsergebnisse zu erlangen. Abbildung nach Cook und Daniels (1994).

erfolgen: die ausschließliche Nutzung von Standard-Programmiersprachen (z. B. FORTRAN, C, C++, Pascal); die zusätzliche Nutzung spezifischer Simulations-Bibliotheken (z. B. SIMPAS für Pascal); die Verwendung spezieller Simulationssprachen (z. B. CSMP); die Nutzung von Simulationsumgebungen (z. B. STELLA)[7]. Die Erstellung des *Simulationsmodells* erfordert neben der Implementierung der Elemente des konzeptionellen Modells die Einführung neuer softwaretechnischer Elemente (beispielsweise um Daten zu lesen und zu speichern oder zur Realisierung der Benutzungsoberfläche). Die Aufgaben bis zur Erstellung des Simulationsmodells werden in Abbildung 2.2 noch einmal verdeutlicht.

Im Schritt der Modellvalidierung wird die Gültigkeit des Modells geprüft: das Systemverhalten wird einer logischen Prüfung unterzogen und die Simulationsergebnisse werden mit evtl. vorhandenen (gemessenen) Daten aus der Realität verglichen. Wenn sich bei der Validierung herausstellt, dass die Simulationsergebnisse die Realität nicht in ausreichendem Maße widerspiegeln, ist eine Überarbeitung des Modellkonzepts vorzunehmen. Sofern das Simulations-

Validierung

[7] Zu den Vor- und Nachteilen der verschiedenen Simulationsverfahren siehe Bossel (1994).

modell freie Parameter enthält, muss eine Anpassung dieser Parameter durch eine Modellkalibrierung erfolgen.

Simulation

Wenn die Modellvalidierung ein zufriedenstellendes Ergebnis geliefert hat, ist der nächste Schritt die Planung und Durchführung von Simulationsläufen (Simulationen). Grundlage für die Simulationsläufe sind i. d. R. so genannte 'Szenarien', die auf „in sich konsistenten und plausiblen Annahmen über die zukünftige Entwicklung systembeeinflussender exogener Größen" (Bossel, 1994) basieren. Weitere Festlegungen betreffen die Auswahl der Anfangszustände des Simulationsmodells sowie die Anfangs- und Endzeitpunkte für die Simulationen (Laufzeitparameter). Bei stochastischen Modellen ist zusätzlich die Anzahl der Simulationsdurchläufe pro Szenario zu bestimmen. Nach der Festlegung und Beschaffung der benötigten Daten und Parameter werden die Simulationsläufe durchgeführt.

Unsicherheitsanalysen

Um die Auswirkungen von Unsicherheiten der Modelleingabegrößen auf die Modellergebnisse bzw. den Einfluss einzelner Eingabegrößen auf die Modellergebnisse zu untersuchen, sollten Unsicherheits- bzw. Sensitivitätsanalysen durchgeführt werden.

Auswertung

Im letzten Schritt zur Durchführung einer Simulationsstudie werden die berechneten Simulationsergebnisse ausgewertet und die Ergebnisse (sofern möglich) umgesetzt bzw. Entscheidungsträgern vermittelt, wobei die Analyse und Dokumentation der Simulationsergebnisse stets im Zusammenhang mit den getroffenen Annahmen zu erfolgen hat.

Ablauf

Die einzelnen Phasen werden in der Regel nicht einfach nacheinander abgearbeitet; eine Änderung der Reihenfolge der Schritte kann ebenso sinnvoll sein wie der Rücksprung zu bereits bearbeiteten Phasen. Beispiele sind die bereits angeführte Notwendigkeit der Überarbeitung des Modellkonzepts im Zuge der Modellvalidierung oder eine Modellkonzeptänderung, wenn sich herausstellt, dass benötigte Daten nicht verfügbar sind.

Eine ausführliche Beschreibung der Schritte findet sich bei Steinhausen (1994). Eine detaillierte Vorgehensweise zur Entwicklung von Modellkonzepten und Simulationsmodellen sowie der Analyse von Modellsystemen liefert Bossel (1994).

2.3 Software-Entwicklung

2.3.1 Entwicklungsphasen

Die Entwicklung eines Software-Systems geschieht in verschiedenen Entwicklungsphasen. Jede Entwicklungsphase hat ihre spezifizierte Aufgabe, verwendet strukturell definierte Eingabedokumente und liefert festgelegte Ergebnisdokumente. An dieser Stelle wird ein Überblick über die einzelnen Phasen der

Software-Entwicklung gegeben, um die Arbeit der Architekturentwicklung einordnen zu können.

Balzert (1996) teilt die Gesamtentwicklung eines Software-Systems in sechs Phasen ein:

1. Planungsphase
2. Definitionsphase
3. Entwurfsphase
4. Implementierungsphase
5. Abnahme- und Einführungsphase
6. Wartungs- und Pflegephase

Die *Planungsphase* beschäftigt sich mit einer Voruntersuchung und einer Durchführbarkeitsstudie unter Berücksichtigung fachlicher, ökonomischer und personeller Aspekte. Ein Ergebnis dieser Phase ist das so genannte Lastenheft – eine Zusammenfassung aller fachlichen Basisanforderungen aus der Sicht des Auftraggebers. In diesem Dokument sollen z. B. das Einsatzgebiet des Software-Systems und die hauptsächlichen Funktionen und Daten beschrieben werden. Besondere Leistungsanforderungen (z. B. an den Datenumfang und die Genauigkeit oder Zuverlässigkeit des Systems) werden ebenfalls in diesem Dokument aufgeführt. Neben dem Lastenheft sind eine Projektkalkulation und ein Projektplan Ergebnis der Planungsphase. Am Ende dieser Phase steht die Durchführbarkeitsstudie und damit die Entscheidung, ob das System entwickelt werden soll oder nicht. Planung

Die *Definitionsphase* beschäftigt sich mit der Analyse und Definition der Anforderungen, die vom Auftraggeber an das System gestellt werden. Durch die Befragung der Auftraggeber und der zukünftigen Benutzer sollen vollständige, konsistente, eindeutige und durchführbare Produktanforderungen definiert werden. Zur Unterstützung der Systemanforderungsanalyse (Systemanalyse) können verschiedene Basiskonzepte der Software-Entwicklung eingesetzt werden (z. B. Datenflussdiagramme, ER-Diagramme[8], Klassendiagramme). Ergebnis der Definitionsphase ist die Produktdefinition, die in der Regel ein verbal beschriebenes Pflichtenheft, ein eher formell beschriebenes Produktmodell und ein Konzept für die Benutzungsoberfläche beinhaltet. Definition

Basierend auf den Ergebnissen der Definitionsphase geht es in der *Entwurfsphase* (Designphase) um die Entwicklung einer softwaretechnischen Lösung für das Software-Produkt. Das Ergebnis dieser Phase ist die *Software-Architektur* des Systems und die Spezifikation der Systemkomponenten. Da diese Phase bei der vorliegenden Arbeit einen Hauptbestandteil ausmacht, wird sie im folgenden Unterabschnitt (2.3.2) separat und ausführlicher als die anderen Phasen betrachtet. Entwurf

[8]Entity-Relationship-Diagramm; werden vornehmlich zur Beschreibung permanent zu speichernder Daten und derer Beziehungen untereinander eingesetzt.

Imple-mentie-rung

Die in der Entwurfsphase spezifizierten Systemkomponenten werden in der *Implementierungsphase* durch Programme realisiert. Ergebnisse der Implementierungsphase sind neben dem ablauffähigen Produkt (Objektprogramm) die Quellprogramme (inklusive integrierter Dokumentation) und Testplanungen mit zugehörigen Testprotokollen.

Abnahme u. Einführung

Innerhalb der *Abnahme- und Einführungsphase* geschieht die Übergabe des Gesamtprodukts (alle Produkte und Teilprodukte der vorausgegangenen Phasen) an den Auftraggeber, die in der Regel auch mit einem Abnahmetest einhergeht. Das Ergebnis der Abnahmephase ist ein Abnahmeprotokoll, in dem die Tests und Ergebnisse dokumentiert werden. Während der Einführungsphase wird das Produkt beim Auftraggeber installiert, die Benutzer werden geschult und das Produkt wird in Betrieb genommen. Die Ergebnisse dieses Schrittes werden im Einführungsprotokoll festgehalten. Ab diesem Zeitpunkt unterliegt das Produkt der Wartung und Pflege.

Wartung

In die *Wartungs- und Pflegephase* fällt die Lokalisierung von Restfehlern, die Optimierung des Produkts sowie das Vornehmen von Anpassungen, Änderungen und Erweiterungen.

Gesamt-prozess

Der Gesamtprozess der *Software-Entwicklung* ist eingebettet in die Aktivitäten des Software-Managements und der Qualitätssicherung, auf die hier nicht eingegangen wird und für die an dieser Stelle z. B. auf Balzert (1998) verwiesen wird. Die zu berücksichtigenden Anforderungen bei der Entwicklung der Software-Architektur sind, aufgrund der Relevanz für die vorliegende Arbeit, ebenfalls im folgenden Unterabschnitt genauer aufgeführt.

2.3.2 Software-Architektur

Der Architektur-Begriff

Ziel der Entwurfsphase ist, wie im vorigen Abschnitt bereits erwähnt, die Entwicklung einer Software-Architektur. Trotz der Wichtigkeit der Architektur in der Software-Entwicklung wird der Begriff in der Literatur unterschiedlich und oft nur vage definiert (Foegen und Battenfeld, 2001). Oestereich (1998) bezeichnet beispielsweise die Spezifikation der *grundlegenden Struktur* eines Systems als Architektur. Nach Buschmann u. a. (1998) beschreibt die Architektur die Subsysteme und Komponenten eines Software-Systems und die Beziehungen zwischen ihnen – die grundlegende Struktur wird in dieser Sichtweise also bereits unterteilt. Die Object Management Group (OMG) definiert in der *Unified Modeling Language Specification* (OMG, 1999) den Begriff der *Architektur* noch detaillierter:

> Architecture: The organizational structure and associated behavior of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect

> parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.

Die Definition der OMG berücksichtigt die wesentlichen Merkmale vieler anderer Definitionen (vgl. Boosch u. a., 1999; Foegen und Battenfeld, 2001; Oestereich, 1998; Poetzsch-Heffter, 2001; Shaw und Garlan, 1996):

- die Aufteilung des Gesamtsystems in mehrere Teile (Subsysteme, Komponenten, Klassen)
- die Kommunikation der einzelnen Teile über Schnittstellen
- die Beziehungen der einzelnen Teile untereinander

Im Rahmen der vorliegenden Arbeit soll der Begriff der 'Software-Architektur' daher wie folgt verstanden werden:

> Eine Software-Architektur ist die grundsätzliche Strukturierung eines Software-Systems. Sie beschreibt eine Menge definierter Komponenten, die über Schnittstellen miteinander kommunizieren, spezifiziert deren jeweiligen Zuständigkeitsbereich und beschreibt die Beziehungen zwischen den Komponenten. *Def.*

Der Begriff der 'Komponente' (auch als 'Halbfabrikat' bezeichnet) soll hier verstanden werden als „... ein abgeschlossener, *binärer* Software-Baustein, der eine anwendungsorientierte, semantisch zusammengehörende Funktionalität besitzt, die nach außen über Schnittstellen zur Verfügung gestellt wird.“ (Balzert, 2000) Komponente

Sichtweisen auf ein System

Die Betrachtung von Software-Systemen kann aus unterschiedlichen Blickwinkeln und auf verschiedenen Abstraktionsniveaus erfolgen. Der Benutzer des Systems ist beispielsweise daran interessiert, *welche* Funktionen die Software bereitstellt; den Entwickler hingegen interessiert *wie* das System diese Funktionen bereitstellt – beide betrachten dasselbe Software-System also aus unterschiedlicher Sicht. Die Betrachtung der Funktionen des Systems kann bei beiden Sichtweisen mehr oder weniger abstrakt sein: der Benutzer kann die Gesamtfunktion betrachten, die das Systems für seine Arbeit hat oder aber die einzelnen Funktionen, die er zur Lösung des Gesamtproblems benötigt; der Entwickler kann (z. B. im Rahmen eines Tests oder der Fehlersuche) eine Funktion als Ganzes untersuchen oder die einzelnen Anweisungen der Funktion.

In der Literatur gibt es unterschiedliche Kategorisierungen von Sichtweisen auf ein System (s. z. B. Foegen und Battenfeld, 2001; Boosch u. a., 1999; Poetzsch-Heffter, 2001; Farooqui u. a., 1995). Eine Einteilung, die auch im Laufe dieser Arbeit eine Rolle spielen wird, ist diejenige des internationalen Standards ISO/IEC 10746, dem so genannten *ISO Reference Model for Open Distributed*

Processing (ISO, 1998; Farooqui u. a., 1995). Das Referenz-Modell unterscheidet fünf Sichtweisen:

- enterprise viewpoint (Beschreibung der Einsatzumgebung des Systems)
- information viewpoint (Beschreibung der Informationen des Systems und deren Verarbeitung)
- computational viewpoint (Beschreibung der funktionalen Zerlegung des Systems)
- engineering viewpoint (Beschreibung der Aspekte zur Verteilung des Systems auf mehrere Rechner)
- technology viewpoint (Beschreibung der zur Implementierung verwendeten Technologien)

Abhängigkeiten

Diese Sichtweisen werden innerhalb des Standards als nicht komplett unabhängig voneinander angesehen; Schlüsselelemente einer Sicht treten in anderen Sichten in untergeordneter Form wieder auf. Dennoch gelten die Sichten als ausreichend unabhängig, um Entwurfsentscheidungen für das Gesamtsystem auf Sichtenebene zu treffen. Die Architektur des Gesamtsystems wird über den kompletten Satz miteinander verbundener Sichten ausgedrückt.

Verwendung

Die vorliegende Arbeit konzentriert sich auf die Entwicklung der Software-Architektur und damit auf den *information viewpoint* und den *computational viewpoint* (vgl. Definition des Architekturbegriffs). Aspekte des *enterprise viewpoint* werden allerdings in der Systemdefinition (Kapitel 4, Seite 69) behandelt und das Kapitel der Realisierung (Kapitel 6, Seite 165) geht auf den *engineering viewpoint* sowie den *technology viewpoint* ein. Die einzelnen Sichtweisen der ISO/IEC 10746 werden aufgrund des Schwerpunktes der Arbeit nicht separat spezifiziert.

ISO/IEC 10746 ist u. a. die Grundlage eines wichtigen Standards im Bereich der Geodatenverarbeitung (ISO 19119). Eine kurze Erklärung der ISO/IEC ds10746 und der einzelnen Sichtweisen findet sich daher an späterer Stelle in dieser Arbeit (Unterabschnitt 3.2.4, Seite 52).

Anforderungen an die Architektur

Bei der Entwicklung der Architektur sind vielfältige Faktoren zu berücksichtigen. Balzert (1996) teilt diese Einflussfaktoren in drei Gruppen ein:

- Einsatzbedingungen
- Grundsatzentscheidungen
- Rand- und Umgebungsbedingungen

Einsatz

Die *Einsatzbedingungen* ergeben sich aus dem Produkteinsatz und bestimmen, ob das Produkt sequentiell oder nichtsequentiell ablaufen soll. Im sequentiellen Fall werden Programmanweisungen (Schritte) hintereinander ausgeführt; im nichtsequentiellen Fall können Anweisungen auch nebenläufig (Bearbeitung mehrerer Schritte unabhängig voneinander), parallel (Bearbeitung

mehrerer Schritte gleichzeitig und unabhängig voneinander) oder in Echtzeit (unter Berücksichtigung von Schritt-Terminierungen und Zeitanforderungen) ausgeführt werden.

Grundsatzentscheidungen

Bei der Entwicklung eines Software-Systems müssen gewisse *Grundsatzentscheidungen* hinsichtlich zu verwendender *Hilfssysteme* getroffen werden. Zu diesen Hilfssystemen und Hilfsdienstleistungen zählt Balzert (1996) Systeme zur Datenhaltung (relationale oder objektorientierte Datenbanken), Hilfesysteme (z. B. Hypertext-basierte Systeme), Expertensystem-Shells und Funktionsbibliotheken zur Erstellung von Benutzungsoberflächen. Das generelle Ziel bei bei Entwicklungsentscheidungen sollte laut Balzert (1996) sein, möglichst viele Dienstleistungen auf hohem Abstraktionsniveau von anderen Systemen in Anspruch zu nehmen.

Bedingungen

Die *Umgebungs- und Randbedingungen* beziehen sich auf die Zielplattform bzw. Zielplattformen, für die das Software-System entwickelt werden soll, also auf die Hardware-, Software- und Anwendungs-Software-Umgebung. Darüber hinaus spielen die *nicht-funktionalen* Anforderungen an das Software-System an dieser Stelle eine wichtige Rolle. Diese Anforderungen beschreiben die *Qualität* eines Systems und sind, im Gegensatz zu den funktionalen Anforderungen, nicht direkt aus der Anforderungsdefinition der Definitionsphase ersichtlich.

nicht-funktionale Eigenschaften

Wichtige nicht-funktionale Eigenschaften sind nach Buschmann u. a. (1998): Änderbarkeit (Wartbarkeit, Erweiterbarkeit, Restrukturierbarkeit, Portierbarkeit), Interoperabilität, Effizienz, Zuverlässigkeit (Fehlertoleranz, Robustheit), Testbarkeit und Wiederverwendbarkeit.

Einen umfassenden Kriterien-Katalog zur Beurteilung von Software-Qualität, der nicht nur bei der Entwicklung der Software-Architektur, sondern auch in den anderen Phasen der Software-Entwicklung berücksichtigt werden sollte, liefert die international Norm ISO/IEC 9126 (übernommen in DIN 66272), die im weiteren Verlauf dieser Arbeit verwendet wird. Eine kurze Erklärung der einzelnen Kriterien dieses Katalogs findet sich in Tabelle 2.1 (Seite 23).

Interoperabilität

Das in der Tabelle 2.1 (Seite 23) aufgeführte Qualitätsmerkmal ‘Interoperabilität’ spielt in dieser Arbeit eine wichtige Rolle. ISO 2382-1 (ISO, 1993)[9] definiert und erklärt den Begriff wie folgt:

Def. Interoperabilität

> Die Fähigkeit zur Kommunikation, Programmausführung oder Datenübertragung über verschiedene funktionale Einheiten hinweg, in einer Art, die vom Nutzer geringe oder keine Kenntnisse der speziellen Charakteristiken dieser Einheiten verlangt. Zwei Komponenten X und Y können *interoperieren* (sind *interoperabel*), wenn X Anfragen R für Dienste an Y senden kann, basierend auf einem gemeinsamen Verständnis von R durch X und Y, und wenn Y entsprechend gemeinsam verständliche Antworten S an X zurückliefern kann.

[9] Zitiert nach ISO/DIS 19119 (Percivall, 2002).

Für Systeme zur Verarbeitung geographischer Informationen gibt es eine weitergehende Definition. Die *'geographische Interoperabilität'* umfasst laut ISO 19119 (Percivall, 2002) zwei Fähigkeiten von Informationssystemen: 1) diejenige zum freien Austausch jeglicher räumlicher Informationen über die Erde und die Objekte und Phänomene auf, über und unter der Erdoberfläche und 2) die Fähigkeit zum (netzwerkbasierten) kooperativen Betrieb von Software zur Manipulation solcher Informationen. ISO 19119 (Percivall, 2002) unterscheidet darüber hinaus noch zwischen *syntaktischer Interoperabilität*, die eine technische Verbindung, also einen Datenaustausch zwischen Systemen sicherstellt, und *semantischer Interoperabilität*, die sicherstellt, dass der Inhalt von beiden Systemen (inklusive interagierenden Personen) gleich interpretiert wird.

Die verschiedenen Gesichtspunkte der Interoperabilität werden in den kommenden Kapiteln noch mehrfach angesprochen.

Merkmal (**Name:**Erklärung)	Teilmerkmal (**Name:**Erklärung)
Funktionalität: Vorhandensein einer Menge von Funktionen und deren festgelegte Merkmale. Die Funktionen sind jene, die die festgelegten oder vorausgesetzten Erfordernisse erfüllen	**Angemessenheit:** Vorhandensein und Eignung einer Menge von Funktionen für spezifizierte Aufgaben
	Richtigkeit: Liefern der richtigen oder vereinbarten Ergebnisse oder Wirkungen
	Interoperabilität: Eignung, mit vorgegebenen Systemen zusammenzuwirken
	Ordnungsmäßigkeit: Erfüllung anwendungsspezifischer Normen, Vereinbarungen, gesetzlicher Vorschriften oder ähnlicher Bestimmungen
	Sicherheit: Eignung, unberechtigten Zugriff, sowohl versehentlich als auch vorsätzlich, auf Programme und Daten zu verhindern
Zuverlässigkeit: Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren	**Reife:** Häufigkeit von Versagen durch Fehlzustände in der Software
	Fehlertoleranz: Eignung, ein spezifisches Leistungsniveau bei Software-Fehlern oder Nicht-Einhaltung ihrer spezifischen Schnittstellen zu bewahren
	Wiederherstellbarkeit: Möglichkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen, unter Berücksichtigung der dafür benötigten Zeit und des benötigten Aufwands
Benutzbarkeit: Aufwand, der zur Benutzung erforderlich ist	**Verständlichkeit:** Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen
	Erlernbarkeit: Aufwand für den Benutzer, die Anwendung zu erlernen
	Bedienbarkeit: Aufwand für den Benutzer bei der Bedienung und Ablaufsteuerung
Effizienz: Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Betriebsbedingungen	**Zeitverhalten:** Antwort- und Verarbeitungszeiten und Durchsatz bei der Funktionsausführung
	Verbrauchsverhalten: Aufwand an Betriebsmitteln bei der Funktionserfüllung
Änderbarkeit: Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist	**Analysierbarkeit:** Notwendiger Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen
	Modifizierbarkeit: Notwendiger Aufwand zur Ausführung von Verbesserungen, zur Fehlerbeseitigung oder zur Anpassung an Umgebungsänderungen
	Stabilität: Risiko unerwarteter Wirkungen von Änderungen
	Prüfbarkeit: Aufwand zur Prüfung der geänderten Software
Übertragbarkeit: Eignung der Software, von einer Umgebung in eine andere übertragen zu werden	**Anpassbarkeit:** Möglichkeit, die Software an verschiedene festgelegte Umgebungen anzupassen, wenn nur Schritte unternommen oder Mittel eingesetzt werden, die für diesen Zweck für die Software vorgesehen sind
	Installierbarkeit: Notwendiger Aufwand zur Installation der Software in einer festgelegten Umgebung
	Konformität: Merkmale, die bewirken, dass die Software Normen oder Vereinbarungen zur Übertragbarkeit erfüllt
	Austauschbarkeit: Möglichkeit und Aufwand die Software anstelle einer spezifizierten anderen Software in der Umgebung jener Software zu verwenden

Tabelle 2.1: Qualitätsmerkmale und Qualitäts-Teilmerkmale der ISO/IEC 9126 (DIN 66272) – Bewertung von Softwareprodukten (DIN, 1994).

Kapitel 3

Stand der Technik

Dieses Kapitel gibt einen Überblick über den Stand der Technik im Bereich der integrierten Modellierung und über Arbeiten, die im Zusammenhang mit der Erstellung der Software-Architektur relevant sind. Ziel

Der erste Abschnitt beschäftigt sich mit Software-Systemen zur integrierten Modellierung. Um den Rahmen der relevanten Systeme abzugrenzen, werden zunächst verschiedene Definitionen des Begriffs 'integriertes Modell' gegenübergestellt. Im Anschluss wird ein kurzer Überblick über existierende Software-Systeme zum 'integrierten simulationsbasierten Assessment' gegeben und es werden einige Frameworks und Entwicklungsumgebungen für derartige Systeme vorgestellt. Da die Wiederbenutzbarkeit und Interoperabilität im Zusammenhang mit integrierten Assessments eine wichtige Rolle spielt, werden in Abschnitt 3.2 (Seite 41) einige Standards vorgestellt, die bei der Entwicklung der Software-Architektur von Interesse sind. Überblick

3.1 Integrierte Modelle

3.1.1 Definitionen

In der Literatur sind – analog zur Definition des integrierten Assessments – verschiedenen Definitionen und Auffassungen des Begriffs 'integriertes (Assessment-) Modell' (IAM)[1] zu finden. Alcamo (2002) charakterisiert integrierte Modelle wie folgt: Definitionen

> Although there is no widely accepted definition of integrated models, most researchers would agree that they: (i) include information

[1] Die Begriffe 'integriertes Modell' (engl. integrated model) und 'integriertes Assessment-Modell' (engl. 'integrated assessment model') werden in dieser Arbeit synonym verwendet.

> from at least two disparate disciplines, (ii) represent this information in the form of discrete programming modules or submodels, and (iii) explicitly or implicitly link scientific findings with policy analysis.

Diese Definition bezieht sowohl das Hauptmerkmal des integrierten Assessments – den Beitrag mehrerer Disziplinen – mit ein als auch den Hinweis auf die Verwendung von Simulationsmodellen sowie die Politikrelevanz. Die Beantwortung politischer Fragestellungen spielt auch bei der Definition von van der Sluijs (1996) eine zentrale Rolle[2]:

> In this paper we define an integrated assessment model as a mathematical representation of a coupled natural system and a socio-economic system, modeling one or more cause-effect chains including feedback loops, and explicitly designed for the purpose of addressing policy questions, mostly by means of scenario analysis.

Die charakteristische Wichtigkeit der Integration verschiedener Fachdisziplinen wird in dieser Definition, laut van der Sluijs (1996), über die Kopplung des natürlichen und sozio-ökonomischen Systems zum Ausdruck gebracht. Im Rahmen des IPCC wird die folgende Definition vorgeschlagen (Weyant u. a., 1996):

> Integrated Assessment Models (IAMs) use a computer program to link an array of component models based on mathematical representations of information from the various contributing disciplines.

Während van der Sluijs (1996) von einer mathematischen Repräsentation des Systems Erde und damit vom konzeptionellen Modell (siehe Abschnitt 2.2, Seite 14) spricht, bezieht sich die Definition von Weyant u. a. (1996) (ebenso wie die von Alcamo, 2002) bereits auf die Umsetzung als Computer-Programm, also auf das Simulationsmodell. Darüber hinaus heißt es: ein IAM *nutzt* ein Computer-Programm, was impliziert, dass IAMs aus mehr bestehen, als aus den einzelnen Teilmodellen. Aus der Sicht der Software-Entwicklung ist diese Unterscheidung durchaus wichtig, da es hiernach Komponenten innerhalb eines IAM geben könnte, die unabhängig von einem konkreten System entworfen und wieder verwendet werden können. Diese Interpretation spiegelt sich auch in der Definition von Rotmans (1998) wider:

> Integrated Assessment models are computer simulation (including optimisation) frameworks that try to describe quantitatively as much as possible of the cause-effect relationship of a specific issue, and of the interlinkages and interactions among different issues.

[2]Über die Politikrelevanz grenzt van der Sluijs (1996) auch die integrierten Assessment-Modelle von den so genannten 'Earth-System Models' ab, die nach seiner Auffassung primär für wissenschaftliche Zwecke erstellt werden.

Ein IAM wird von Rotmans also als ganzes ‘Framework’[3] zur Quantifizierung von Ursache-Wirkungs-Ketten gesehen und beinhaltet demnach mehr als die reinen Simulationsmodelle. Der zusammenfügende Charakter eines Frameworks spielt auch bei der Definition von Toth (1995) eine Rolle:

> In this paper, and throughout the collection that follows, the terms ‘integrated model’ and ‘integrated assessment’ refer to a set of formal models or studies without modeling support that are combined into a consistent framework to address one or more issues in the problem of global climate change.

Wie zu sehen ist, gibt es unterschiedliche Blickwinkel auf ein integriertes Modell: es kann – wie bei van der Sluijs (1996) und Toth (1995) – vom konzeptionellen Standpunkt aus gesehen werden oder – wie bei den anderen Beispielen – als Simulationsmodell. Bei der Betrachtung des Modells kann darüber hinaus noch unterschieden werden zwischen dem ‘reinen’ Modell oder einer Umgebung (Framework), in die das Modell eingebettet ist. Blickwinkel

Im Rahmen dieser Arbeit wird ein integriertes Modell als die softwaretechnische Realisierung des konzeptionellen Modells, also als Software-System, angesehen. Wie in den folgenden Unterabschnitten zu sehen sein wird, bestehen die Software-Systeme integrierter Modelle i. d. R. aus mehr als nur den gekoppelten Simulationsmodellen. Um dies zu verdeutlichen und um den integrierenden Charakter des Gesamtsystems hervorzuheben, wird fortan statt von ‘integrierten Modellen’ von *‘Systemen zum simulationsbasierten integrierten Assessment’ (SISAs)* gesprochen. In Anlehnung an den Begriff des ’integrierten Assessments’ (s. S. 13) soll der Begriff SISA wie folgt verstanden werden: SISA

> Ein System zum integrierten simulationsbasierten Assessment (SISA) ist ein Software-System, das von unterschiedlichen Fachdisziplinen stammende Daten und Simulationsmodelle zum ‘System Erde’ in einem konsistenten Rahmen kombiniert und neue Daten über den Zustand und mögliche langfristige Änderungen des ‘Systems Erde’ – vornehmlich zur Unterstützung politischer Entscheidungsträger – berechnet und bereitstellt. *Definition*

Neben der Bezeichnung ‘SISA’ wird der Begriff des ‘integrierten Modells’ im weiteren Verlauf der Arbeit dann benutzt, wenn die gekoppelten Simulationsmodelle im Mittelpunkt stehen.

[3] Rotmans führt als Beispiele für IAM die ablauffähigen Systeme RAINS (Alcamo u. a., 1990) und IMAGE (Alcamo, 1994) an. Aus diesem Grund kann davon ausgegangen werden, dass er die Bezeichnung ‘Framework’ für ein weit gefasstes System verwendet und nicht – wie aus der Sicht der Informatik üblich – für ein auszubauendes Rahmenwerk zur Entwicklung lauffähiger Systeme (vgl. ‘Framework’-Definition im Glossar, Seite 229).

Archi-tektur

Zur Entwicklung einer Software-Architektur für ein SISA müssen über eine System-Definition u. a. die wichtigsten Ziele und Funktionen sowie die System-Umgebung definiert werden. Ausgangspunkt für die System-Definition und die Architektur-Entwicklung sind Veröffentlichungen integrierter Modelle, die Aufschluss darüber geben sollen, welche generellen Funktionen, Ziele und unterstützenden Programme zu berücksichtigen sind. Die folgenden Unterabschnitte geben einen Überblick über diese Anforderungen und geben gleichzeitig Informationen zur Strukturierung aktueller Systeme.

3.1.2 Systeme

Überblick

Kickert u. a. (1999) liefern einen Statusbericht über Simulationsmodelle zur Evaluierung möglicher ökologischer, umweltbezogener und sozialer Konsequenzen des globalen Wandels.[4] Der Bericht beschreibt eine sehr umfangreiche Auswahl unterschiedlichster (integrierter) Modelle. Die Beschreibungen der einzelnen Modelle beschränken sich in den meisten Fällen auf Angaben zu den *konzeptionellen* Modellen in Form von Flussdiagrammen. Erklärungen zu softwaretechnischen Aspekten finden sich i. d. R. nicht – und wenn, dann unter Verwendung uneinheitlicher Darstellungsformen und unterschiedlichster Abstraktionsniveaus: angefangen von der sehr groben Einteilung eines Modells in drei Komponenten (Model, Control, Show) über Objekte eines objektorientiert programmierten Systems bis hin zu Dateistrukturen. Angaben über den *grundsätzlichen* Aufbau gesamter Software-Systeme sind hier nicht zu finden.

offener Quell-code

Die Autoren sprechen in diesem Bericht allerdings die Wichtigkeit der Offenheit und freien Verfügbarkeit des Quellcodes von Simulationsmodellen an, da Ergebnisse nur über diesen Weg von Dritten reproduzierbar seien. Als weiteren wichtigen Punkt bezüglich der Reproduzierbarkeit von Ergebnissen nennen sie die Versionierung von Programmen.

Anforde-rungs-über-sicht

Peirce (1998) liefert in seinem Bericht über ‘Computer-basierte Modelle im integrierten Umwelt-Assessment’ Kurzbeschreibungen von insgesamt 27 Software-Systemen, die im Rahmen des integrierten Assessments benutzt werden. Diese Liste enthält unterschiedliche Systeme, angefangen von Simulationsmodellen über Entwicklungsumgebungen bis hin zu GIS. Auf die einzelnen Systeme soll an dieser Stelle nicht eingegangen werden. Die Analyse der aufgeführten Systeme zeigt allerdings das Funktionsspektrum, das im Rahmen des integrierten Assessments benötigt wird und das an dieser Stelle nur stichpunkthaft dargestellt werden soll: Visualisierung von Ergebnissen, Datenbank-Schnittstellen, GIS-Funktionen, Datenanalyse, Optimierung, Graphische Benutzungsschnittstelle (GUI), Unsicherheitsmodellierung (Latin Hypercube, Monte-Carlo Si-

[4] Der Bericht legt seinen Schwerpunkt allerdings auf die ökologischen Aspekte des globalen Wandel.

mulation), Statistische Berechnungen, Internet-Zugriff von Informationen, Unterstützung bei der Verfassung von Berichten, Datenbasis, Wissensbasis mit Checklisten, Regeln, Hintergrundinformationen für den Analysten, Inferenz-Maschine[5], Summary-Report-Generator, Animation, GUI-Design Tool, Integriertes Web-Publishing, Zeitserien-Analyse, Hilfesystem.

Nicht jedes System zur Unterstützung des integrierten Assessments muss all diese Funktionalitäten bereitstellen. Die Auflistung zeigt allerdings das Leistungsspektrum der verwendeten Systeme.

Architekturen

Aufschlussreiche Informationen über einen grundsätzlichen Aufbau von SISAs, also der Software-Architektur solcher Systeme, finden sich bei Peirce (1998) ebenso wenig wie bei Kickert u. a. (1999). Dieses Phänomen kann nicht alleine darauf zurückzuführen sein, dass die zugrunde liegenden Software-Konzepte im Rahmen derartiger Berichte nicht von Interesse wären – die Beschreibungen von Programmobjekten und Dateistrukturen zielen schließlich auf die softwaretechnische Umsetzung. Die Ursache für fehlende Beschreibungen von Software-Architekturen in solchen Berichten ist vielmehr darin zu suchen, dass die Architektur-Konzepte – falls überhaupt vorhanden – nicht bzw. nicht für die Öffentlichkeit dokumentiert sind. Selbst in ausführlichen Beschreibungen wie den Büchern zu den integrierten Modellen IMAGE2 (Alcamo, 1994) und AIM (Kainuma u. a., 2003) finden sich keine ausdrücklichen Beschreibungen der Architektur-Konzepte. Die angeführte Literatur stützt die Aussagen von Jaeger u. a. (2002), dass integrierte Modelle i. d. R. nur unzureichend dokumentiert und modularisiert sind.

Wie aufschlussreich eine Beschreibung der Teile des Gesamtsystems sein kann, wird an der 'Architektur'-Beschreibung des Systems 'GLOBESIGH' gezeigt – eine der wenigen Beschreibungen dieser Art.

Beispiel GLOBESIGHT

Mesarovic u. a. (1996) stellen ein 'integrated assessment support system' namens *GLOBESIGHT (global insight)* vor. Das System besteht aus vier Komponenten (Abbildung 3.1): der *information base*, der *model (algorithms) base*, der *functionalities base (tools base)* und der *issues base* (s. Abb. 3.1, Seite 30).

information base

Die *information base* enthält Daten und Informationen, die während der Untersuchung einer bestimmten Fragestellung für den Nutzer hilfreich sind. Hierzu können z. B. textuelle Hintergrundinformationen über die Geographie oder sozio-ökonomische Daten (Zeitreihen der Bevölkerungszahlen, des Bruttosozialprodukts, des Ressourcen-Verbrauchs etc.) eines Landes oder einer Region gezählt werden.

[5] In wissensbasierten Systemen benutzt, um aus Fakten und Regeln neues Wissen abzuleiten.

model base

Die *model base* enthält Modelle einzelner Sub-Systeme (vergleichbar mit den Sphären aus Kapitel 2.1.1): Simulationsmodelle zur Bevölkerung/Demographie, zur Ökonomie und zum Ressourcen-Verbrauch sind hier beispielsweise vorhanden. Über Szenarien können mit Hilfe der in der Modell-Basis enthaltenen Simulationsmodelle mögliche zukünftige Entwicklungen und Konsequenzen politischer Entscheidungen analysiert werden. Je nach Fragestellung und betrachteter Region bzw. betrachteten Regionen können für ein Sub-System Modelle mit unterschiedlichem Abstraktionsniveau ausgewählt werden. So stehen z. B. drei unterschiedliche Modelle für den Bereich der Bevölkerungsentwicklung bereit (angefangen von einer einfachen Wachstumsrate über die Gesamtbevölkerung bis hin zu einem Kohorten-Modell, das Geburten- und Sterblichkeitsraten berücksichtigt). Darüber hinaus können Simulationsergebnisse für die nationale, regionale oder globale Ebene berechnet werden.

tools base

Die *functionality (tools) base* stellt Funktionen zur Dateneingabe, Datenausgabe und zur Datenverarbeitung zur Verfügung. Sie unterstützen die Modellverwaltung, den Datenbank-Import und -Export und die Anzeige von Daten über Diagramme und Karten. Darüber hinaus bietet die *functionality (tools) base* Routinen zur Interpolation von Eingabedaten sowie zur Erreichung vorgegebener Ziele (z. B. Emissionsziele).

issue base

Die *issue base* ist eine Ansammlung aller Analysen und enthält sowohl die Ergebnisse als auch die zugrunde liegenden Szenario-Annahmen. Diese Informationssammlung dient als Referenz und Ausgangspunkt für zukünftige Analysen.

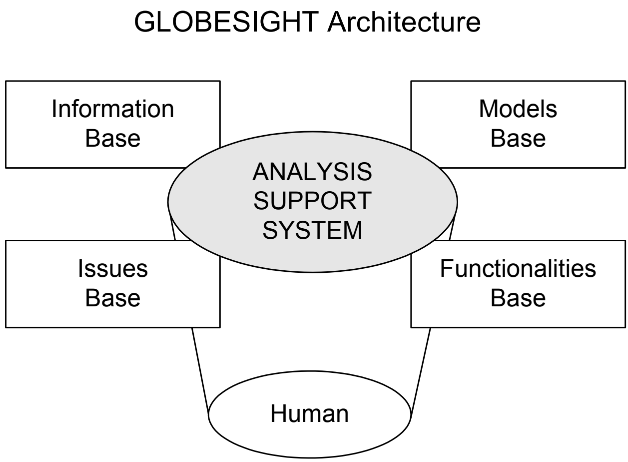


Abbildung 3.1: Architektur des Systems *GLOBESIGHT*. Erklärungen finden sich im Text. Quelle der Abbildung: http://genie.cwru.edu/globesight.htm.

GLOBESIGHT verfolgt einen interaktiven Ansatz und ermöglicht es, z. B. im Rahmen von Workshops, Planspiele durchzuführen: Szenarien werden dabei nicht zu Beginn der Simulation, sondern interaktiv bestimmt – nach einer kurzen Simulation erfolgt eine Rückfrage bei den anwesenden Experten. Diese Vorgehensweise erlaubt es, Entscheidungsträger mit in den Erzeugungsprozess von Simulationsergebnissen einzubeziehen. interaktiver Ansatz

Aufgrund fehlender Gesamtbeschreibungen integrierter Modelle (bzw. SISAs) wird im Folgenden auf Veröffentlichungen über die Struktur von Frameworks und Entwicklungsumgebungen zurückgegriffen.

3.1.3 Frameworks

Object Modeling System (OMS)

Das *Object Modeling System (OMS)* (Busch u. a., 2002) ist ein Framework zur interaktiven Entwicklung und Anwendung dynamischer Simulationsmodelle. Das System ist eine Gemeinschaftsentwicklung vom U.S. Geological Survey (USGS), dem U.S. Department of Agriculture (USDA) und der Friedrich-Schiller-Universität in Jena.

OMS ist modular aufgebaut (s. Abb. 3.2, Seite 32): die Funktionen zur Abbildung des konzeptionellen Modells sind getrennt von den so genannten 'Basisfunktionen'. Über die Basisfunktionen werden z. B. die Dateneingabe und -ausgabe, die Kommunikation zwischen Modellteilen und die Anwendung einzelner Modellkomponenten implementiert. OMS bietet den Entwicklern den Vorteil einer einheitlichen Programmierschnittstelle (engl. Application Programming Interface, API). Modellnutzern wird über OMS eine einheitliche Benutzungsschnittstelle (engl. User Interface, UI) bereitgestellt.

OMS besteht aus *Systemkomponenten* und *Modellkomponenten*. Zu den Systemkomponenten gehören: der Systemkern, der Modellersteller, der Skript-Interpreter, der GIS-Client, das Anwendungs-Framework, das Update-Center und Komponenten für die Benutzungsschnittstelle.

Der Systemkern bietet *Basisfunktionalitäten* für andere Komponenten und ist die Laufzeit-Umgebung für die Entwicklung und Ausführung von Modellen und Komponenten. Der Systemkern stellt auch einfache und komplexe *Datenobjekte* bereit. Diese Datenobjekte können neben dem Wert zusätzliche Informationen halten: einen Namen für das Objekt, die Einheit des Wertes sowie den erlauben Zahlenbereich. Alle OMS-Datentypen besitzen Funktionen zum Lesen und Schreiben der Werte. Über ein integriertes Funktionspaket zur Einheitenverarbeitung[6] ist es möglich, die Kompatibilität von Variablen innerhalb von Formeln zu prüfen und Werte von einer Einheit in eine andere, kompatible umzurechnen. Zur Integration neuer Modellkomponenten müssen deren Systemkern

[6] Integriert wurde das UCAR.UNITS Package der University Corporation for Atmospheric Research. Internet-Startseite: http://www.ucar.edu

Schnittstellen bestimmten Konventionen entsprechen. Die Entwicklung neuer Modellkomponenten wird durch vorgefertigte *Oberklassen* erleichtert, die ebenfalls Teil des Systemkerns sind.

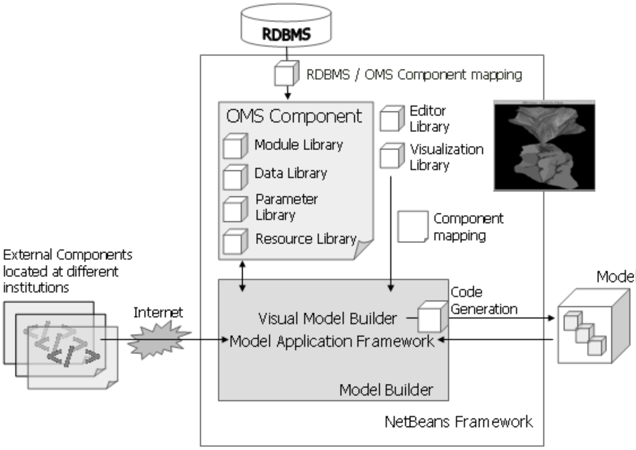


Abbildung 3.2: Architektur des *Object Modeling System (OMS)*. Quelle: Busch u. a. (2002).

Modellbilder

Aufgabe des Modellbilders ist es, die einzelnen OMS-Komponenten zu einem komplexen Modell zusammenzufügen. Über eine GUI-Komponente können z. B. die Modellausgaben einer OMS-Komponente mit den Eingaben einer anderen Komponente verbunden und verschiedene Modellkonfigurationen erstellt und verwaltet werden.

Skript-Interpreter

Die Modellentwicklung und -anwendung wird unterstützt durch einen integrierten Skript-Interpreter. Über diesen Interpreter können z. B. Modelle oder Modellkomponenten gesteuert werden. Eine Interpreter-Konsole zur Ausführung von Befehlen und Änderung von Variablenwerten während der Simulation ist ebenfalls integriert. Als Skript-Sprache wurde Python gewählt.

GIS-Client

Räumliche Daten können mit dem GIS-Client bearbeitet und visualisiert werden. Neben einer Benutzungsschnittstelle stellt der GIS-Client eine Programmierschnittstelle bereit, die über den Python-Interpreter ansprechbar ist.

NetBeans

Als Anwendungsplattform wird das *Open-Source-Produkt*[7] *NetBeans* benutzt. NetBeans[8] ist eine modulare, auf Standards basierte integrierte Entwick-

[7] Die Lizenzen von Open-Source-Produkten gewähren grundlegende Rechte. So dürfen Open-Source-Programme beispielsweise weitergegeben werden und Quelltexte dürfen analysiert und auch in geänderter Form weitergegeben werden. Einen Überblick über die wichtigsten Open-Source-Lizenzen geben Roehrl und Schmiedl (2002).

[8] Startseite im Internet: `http://www.netbeans.org`

lungsumgebung. Das Framework ist in der Programmiersprache JAVA implementiert und unterstützt zurzeit auch nur die Entwicklung mit dieser Sprache.

Update Center

Über das so genannte *Update Center* können existierende Komponenten aktualisiert und neue Komponenten in das OMS integriert werden. Die OMS-Komponenten müssen für diesen Zweck als NetBeans-Module gekapselt sein.

OMS stellt Komponenten für die Benutzungsschnittstelle bereit, die z. B. Daten-Visualisierungsmöglichkeiten in Form von Diagrammen bereitstellen. Die Integration weiterer Komponenten ist in Form von NetBeans-Modulen möglich.

Modellkomponenten

Die OMS-Modellkomponenten sind die Bausteine aller Modelle, die mit dem Framework erstellt werden. Der Prototyp einer solchen Komponente ist Teil von OMS – er gibt die Methoden vor, die implementiert werden müssen: register, init und run. Die register- und init-Methoden enthalten Anweisungen, die zur Initialisierung benötigt werden. Die run-Methode enthält die Anweisungen der eigentlichen Funktionalität des Moduls, die bei jedem Aufruf des Moduls ausgeführt werden.

Die Modellkomponenten sind als JAVA-Klassen implementiert. Die Implementierung der oben genannten Funktionen kann aber nicht nur in JAVA erfolgen; der Code anderer Sprachen, wie beispielsweise FORTRAN oder C++, kann automatisch über das *JAVA Native Interface (JNI)* in die Module integriert werden.

PRISM

Ziel

PRISM (programme for integrated earth system modelling) ist ein Infrastrukturprojekt für die Klima- und Erdsystem-Forschung in Europa.[9] Ziel des von der Europäischen Union geförderten Programms ist die Errichtung eines verteilten europäischen Netzwerks für die Erdsystem-Modellierung. Um dieses Ziel zu erreichen, will PRISM

- eine europäische Dienstleistungs- und Verwaltungsstruktur zur Entwicklung, Koordination und Durchführung langfristiger, europaweiter und multi-institutionaler Klima- und Erdsystem-Simulationen aufbauen
- ein europäisches System portabler, leistungsfähiger und benutzungsfreundlicher Modelle aus dem Erdsystem- und Klimabereich und damit zusammenhängender Diagnose- und Visualisierungs-Software unter standardisierten Codierungskonventionen entwickeln, die für alle europäischen Wissenschaftler zugänglich sind

Ein erwartetes Produkt der Aktivitäten ist eine flexible, effiziente, portable und benutzungsfreundliche Infrastruktur für die Modellierung des Erdsystems und die Klimavorhersage.

[9] Startseite im Internet: http://www.prism.enes.org

Schwerpunkte

PRISM konzentriert sich auf die Modellierung des Erdsystems und zielt auf den vermehrten Einsatz von Super-Computern für die Simulationsberechnungen. Die Komponenten haben die folgenden Schwerpunkte: Atmosphäre, atmosphärische Chemie, Landnutzung, Ozean, Meeres-Eis, marine Bio-/Geochemie, regionale Klimamodelle. Abbildung 3.3 zeigt die schematische Modellstruktur von PRISM. Die explizite Berücksichtigung sozioökonomischer Faktoren und Auswirkungen durch eine separate Komponente ist hier nicht zu finden.

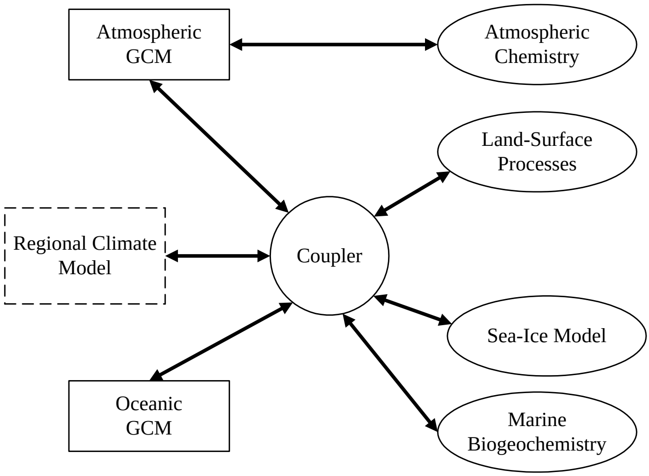


Abbildung 3.3: Schematische Modellstruktur von *PRISM*.

Modellkoppler

Die Modellteile werden über eine standardisierte Schnittstelle und einen *Modellkoppler* interagieren. Als Koppler wird *OASIS (Ocean Atmosphere Sea Ice Soil)* verwendet, der durch das CERFACS[10] bereitgestellt wird. OASIS wurde 1991 vom CERFACS-Team *Klimamodellierung und globaler Wandel* entwickelt um existierende *global circulation models (GCMs)* zu koppeln. OASIS besitzt vier Hauptaufgaben:

1. Modell-Synchronisation
2. Modell-Kopplung
3. Datenaustausch[11]

[10] European Centre for Research and Advanced Training in Scientify Computation.

[11] Zur Synchronisation und zum Datenaustausch werden vier verschiedene Kommunikationswege bereitgestellt: (1) Pipes (CRAY pipes) zur Synchronisation der Modelle und des Datenaustauschs über Binärdateien, (2) die so genannte CLIM-Technik zur Synchronisation und zum Datenaustausch unter Nutzung der Message-Passing-Standards *PVM (Parallel Virtual Machine)* und *MPI (Messaga Pasing Interface)* und (3/4) SIPC (basierend auf Unix V Inter Process Communication) bzw. GMEM (basierend auf NEC global memory concept),

4. Interpolation

OASIS ist stark auf die Möglichkeiten der Parallelisierung und die Architektur von Super-Computern bzw. Skalar-Computern ausgerichtet und nutzt zur Effizienzsteigerung alle verfügbaren Betriebssystem-Mittel aus (z. B. von den Modellen gemeinsam genutzte Speicherbereiche zum Datenaustausch). Die Kopplung findet also auf einem relativ niedrigen Abstraktionsniveau statt. Technik

Integrating Modelling Architecture (IMA)

Die *Integrating Modelling Architecture (IMA)* (Villa, 2001) verfolgt die Integration unterschiedlicher Modellierungsparadigmen[12] und besteht aus einem deklarativen Framework und einem Paket von Software-Werkzeugen (Integrated Modelling Toolkit, IMT).

Das grundlegende Element innerhalb des IMT ist das ‘Modul’. Ein solches Modul (Objekt einer zuvor spezifizierten Klasse) greift bei seiner Ausführung (z. B. der Berechnung von Gleichungen) auf einen gemeinsamen Datenraum zu, wobei der Datenraum wiederum als Modul (Datenmodul) spezifiziert ist. Um die Verbindung von Modulen zu vereinfachen, erhält jedes Modul eine interne Beschreibung (‘DNA’) seiner eigenen Struktur. Für die Beschreibung der Struktur wird die *Extensible Markup Language (XML)* genutzt. Das IMT sieht u. a. Module vor für die Simulationsunterstützung, für Optimierungsrechnungen, die Integration von GIS[13] und Analysewerkzeugen, den Import und Export von Daten sowie die Visualisierung. Module

Die Integration von Teilmodellen, die sich nicht direkt in das System einbetten lassen, soll über Funktionen realisiert werden, die Teilmodelle als externe Programm starten, deren Zeitschritt fortschreiben und Variablen des Teilmodells abfragen. Zu diesem Zweck wird die Nutzung weiterer Software-Werkzeuge, wie dem ‘Simulation Network Interface’ (s. u.), vorgeschlagen. Integration

Simulation Network Interface (SNI)

Das *Simulation Network Interface (SNI)* (Villa und Costanza, 2000) ist ein Software-Paket zur netzwerkbasierten Integration von Simulationsmodellen. Beim Design von SNI wurde Wert darauf gelegt, dass für die Modellkopplung nur ein *geringer Implementierungsaufwand* seitens der zu integrierenden Teilmodelle notwendig ist – im einfachsten Fall der Modellkopplung ruft ein koordinierendes Programm ein als ausführbares Programm vorliegendes Teil- Kommandozeilen-Ansatz

die die Synchronisation und den Datenaustausch über Semaphoren und gemeinsam genutzte Speicherbereiche erreichen.

[12] Genannt werden bei Villa (2001) z. B.: prozessbasierte vs. agentenbasierte, nichträumliche vs. räumliche explizite, deterministische vs. stochastische Modellierung.

[13] Geplant ist die Integration der GIS ArcInfo und GRASS.

modell über die Kommandozeile auf und wertet dessen Bildschirmausgaben als Simulationsergebnis aus (Kommandozeilen-Ansatz).

Master vs. Slave

SNI unterscheidet zwischen *'master'-Anwendungen*, die für die Planung und Koordination einer Simulation zuständig sind, und *'slave'-Simulationsprogrammen*, die durch entfernte Aufrufe über die Kommandozeilen-Schnittstelle kontrolliert werden. Während der 'master' die Funktionsbibliotheken von SNI benutzt, müssen auf der 'slave'-Seite keinerlei Anpassungen vorgenommen werden – abgesehen von der Erweiterung um eine Kommandozeilen-Schnittstelle, falls eine solche nicht vorhanden ist.

Client vs. Server

Implementiert wird SNI durch zwei Software-Komponenten: dem *SNI-Server* und dem *SNI-Client*. Auf allen Hosts, auf denen ein Simulationsmodell laufen soll, muss der SNI-Server gestartet werden. Das 'master'-Programm (die koordinierende Instanz) spricht die einzelnen Simulationsmodelle dann über ein eigens definiertes Protokoll und mit Hilfe des SNI-Client an.[14] Dem SNI-Server muss hierbei lediglich mitgeteilt werden, wie er ein Simulationsmodell ausführen, initialisieren und einen Simulationslauf starten kann.

Beispiel-Dienste

Die Simulationsmodelle können über die Verbindung SNI-Server/SNI-Client auch *andere Server ansprechen* und mit ihnen Daten austauschen. Auf diese Weise können verschiedene Dienste angeboten werden – als Beispiele werden ein *Daten-Server* angeführt, ein Server, der *GIS-Funktionen* zur Verfügung stellt sowie ein Dienst zur *Kalibrierung* von Simulationsmodellen. Bei der automatischen Modell-Kalibrierung werden die Parametersätze unter Angabe eines eindeutigen Bezeichners gespeichert, so dass für spätere Simulationen darauf zurückgegriffen werden kann.

DANUBIA

DANUBIA (Hennicker u. a., 2003) ist ein integriertes Simulationssystem, das im Rahmen des Projektes *GLOWA-Danube*[15] entwickelt wird.

Framework

Das dem System zugrunde liegende objektorientierte Framework basiert auf Internet-Technologien und ist in JAVA realisiert. Im Zentrum eines Systems steht der *Time Controller*, der für die Koordination von Teilmodellen, die auch unterschiedliche Zeitschritte verwenden können, verantwortlich ist.

Datenaustausch

Die Teilmodelle können untereinander über definierte Schnittstellen Daten austauschen. Durch die Implementierung eines Zustands-Übergangs-Modells wird dabei sichergestellt, dass nur auf gültige Werte anderer Teilmodelle zugegriffen werden kann.[16]

Verteilung

Die Kommunikation zwischen Teilmodellen wird über *JAVA Remote Method Invocation (RMI)* abgewickelt. Auf diese Weise können sowohl die Teilmodelle

[14] Zur Verbindung von Server und Client wird das Internet-Protokoll (TCP/IP) genutzt.

[15] Startseite im Internet: `http://www.glowa-danube.de`

[16] 'Gültig' heißt, dass sich die abgefragten Daten nicht in einem undefinierten, gerade in der Berechnung begriffenen Zustand befinden.

als auch der Time Controller auf unterschiedlichen Hosts (Rechnern) laufen. Durch die Nutzung von Adaptern (wrappers) werden die technischen Details der Netzwerk-Kommunikation vor den Teilmodellen versteckt. Abbildung 3.4 zeigt als Beispiel die Architektur eines Systems mit zwei Teilmodellen.

In der nächsten Phase des Projektes sollen weitere Komponenten zur Entscheidungsunterstützung zur Verfügung gestellt werden. Hierzu gehören Funktionen zur Erstellung, Beurteilung und Verwaltung komplexer Szenarien. Darüber hinaus ist eine Web-basierte Benutzungsschnittstelle vorgesehen. Erweiterung

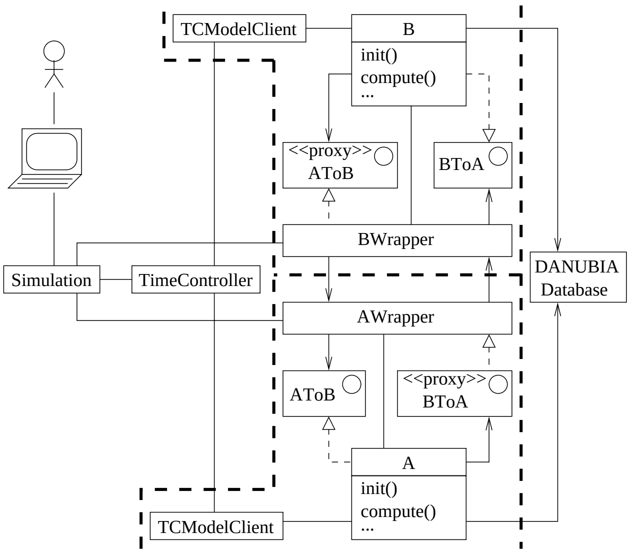


Abbildung 3.4: Systemarchitektur von *DANUBIA*. Beispiel eines Systems mit zwei Teilmodellen (A und B) und einem gemeinsamen 'Time Controller' zur Koordinierung der Zeitschritte. Der Time Controller und die beiden Teilmodelle können auf unterschiedlichen Hosts laufen. Die Kommunikation der Teilmodelle geschieht über definierte Schnittstellen (AToB, BToA). Durch die Verwendung von Adaptern (A/B Wrapper) ist die Netzwerk-Kommunikation für die Modelle transparent. Zur Integration eines Teilmodells muss der Entwickler die Methoden *init()* und *compute()* realisieren. Während der Initialisierung der Modelle greifen diese beispielsweise auf die für das Gesamtsystem konsistenten Daten innerhalb der *DANUBIA Database* zu. Die über den *TimeController* gesteuerten *TCModelClients* veranlassen die Berechnung neuer Ergebnisse. Quelle der Abbildung: Hennicker u. a. (2003).

3.1.4 Entwicklungsumgebungen

Modellentwicklungssystem M

Im Rahmen der Entwicklung und Anwendung komplexer integrierter Modelle in den Bereichen Gesundheitswesen und Klimawandel wurde am *staatlichen Institut für Gesundheit und Umwelt der Niederlande (RIVM)*[17], mit Unterstützung der niederländischen *Energy Research Foundation* ein Werkzeug mit dem Namen *M* entwickelt (de Bruin u. a., 1996).[18] M ist eine integrierte Software-Umgebung zur Entwicklung, Visualisierung und Anwendung von interaktiven, dynamischen Modellen, die auf algebraischen Gleichungen, Differenzengleichungen oder gewöhnlichen Differenzialgleichungen basieren.

Prinzip

Das Prinzip des Systems ist die klare Trennung zwischen dem mathematischen Modell, den Lösungsmethoden, den Daten, der Datenverwaltung und der Benutzungsschnittstelle. Dem Modellentwickler soll dadurch die Möglichkeit gegeben werden, sich auf die Spezifikation der Gleichungen, der Eingabedaten und der Modell-Dokumentation zu konzentrieren.

Komponenten

Die M-Umgebung besteht nach de Bruin u. a. (1996) aus neun unabhängigen Komponenten (s. auch Abb. 3.5):

- dem *M compiler* zur Übersetzung von Gleichungen in ausführbare Programme bzw. Objektdateien (den so genannten Simulatoren)
- dem *visualizer* zum Entwurf und zur Nutzung interaktiver Präsentationen (in Abbildung 3.5 als *graphical interface* bezeichnet)
- dem *command line interface* zum Simulator als Schnittstelle für Testzwecke und zur Ausführung vordefinierter Aufgaben, der so genannten *batch jobs*
- dem *tracer* zur Verfolgung von Abhängigkeiten zwischen Variablen
- dem *table editor* zur alphanumerischen Bearbeitung mehrdimensionaler Daten (nicht in Abbildung 3.5 aufgeführt)
- dem *scenario manager* zum Sichern und Vergleichen verschiedener Modell-Eingaben (scenarios)
- dem *documentation system* zur HTML-basierten Modelldokumentation
- dem *online help system* zur Bedienung und zum Verständnis des Modells (HTML browser)
- dem *application programmers interface* zum direkten Zugriff auf den Simulator (API)

[17] Rijksinstituut voor Volksgezondheid en Milieu. Startseite im Internet: `http://www.rivm.nl`

[18] Startseite des Projektes im Internet: `http://www.m.rivm.nl`.

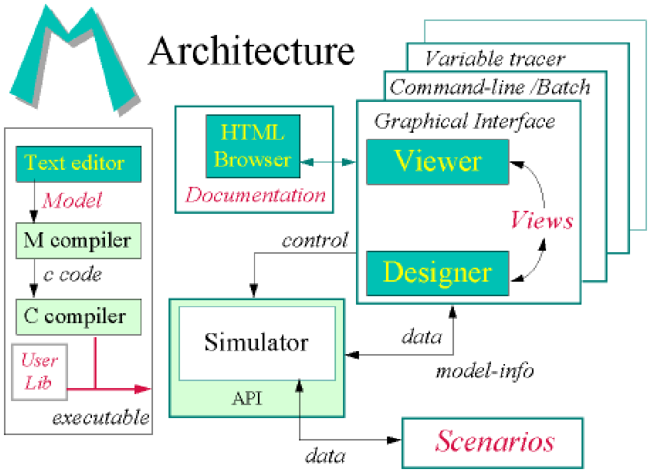


Abbildung 3.5: Architektur der Modellumgebung *M*. Erklärungen finden sich im Text. Quelle: de Bruin (1996).

Die Visualisierungs-Komponente wird von de Bruin u. a. (1996) als die entscheidende Komponente für die Entwicklung eines integrierten Modells angesehen und erfüllt daher auch vielfältige Funktionen: von der dynamischen Visualisierung der modellierten Phänomene mit Diagrammen und Karten und dem Vergleich der Ergebnisse mehrerer Szenarien über direkte Manipulationen von Modell-Daten innerhalb der graphischen Visualisierungen (unter Nutzung des Model-View-Controller-Musters[19]) bis hin zur strukturierten Darstellung von Graphen für alle Modell-Variablen reichen die Möglichkeiten dieser Komponente. Visualisierung

Zur Erzeugung eines ausführbaren Modells muss ein bestimmtes Verfahren eingehalten werden: Die Gleichungen eines Modells sind in einer nicht-prozeduralen mathematischen Sprache (der M-Programmiersprache) zu beschreiben und über einen einfachen Text-Editor im ASCII-Format abzuspeichern. Die so spezifizierten Modelle sind dann mit Hilfe eines Konverters in die Programmiersprache C zu übersetzen und anschließend mit einem handelsüblichen C-Compiler zu kompilieren. Die resultierenden Objekt-Dateien können dann entweder direkt in andere Programme eingebunden oder mit einer graphischen bzw. alphanumerischen Benutzungsoberfläche zu einem ausführbaren Programm verbunden werden. Die Definition der Benutzungsschnittstelle ist Modellerstellung

[19]Beim Model-View-Controller-Muster (MVC-Muster) werden die Funktionen zur Visualisierung von denen zur Manipulation der Daten getrennt; zum MVC-Muster s. Gamma u. a. (1996) oder Buschmann u. a. (1998).

durch so genannte ‘view definition files’ zu beschreiben. Die Prozeduren zur numerischen Integration, Datenverwaltung und Datenspeicherung werden beim Linken automatisch hinzugefügt.

Pipes

Unter UNIX ist es auch möglich, den Simulator und die GUI als einzelne Prozesse zu starten. Die Kommunikation der Prozesse erfolgt dann über den ‘Pipe’-Mechanismus von UNIX.[20] Nähere Informationen zu dieser Art der Simulator-GUI-Verbindung finden sich bei van Wijk (1994).

Modularisierung

Um auch umfangreiche Modelle mit M verwalten zu können, erlaubt M die Aufteilung des Gesamtsystems in Module sowie die Definition von Makros, die Mehrfachverwendung von Dateien und die Integration von C- und FORTRAN-Funktionen. Die bereits oben angesprochene Verwaltung der Daten in unterschiedlichen Dateien (zur Beschreibung der Modellgleichungen, der Nutzungsschnittstelle und der unterschiedlichen Prozeduren) erleichtert ebenfalls die Handhabung eines in M entwickelten Modells.

Einsatz

Seit der Geburtsstunde des Systems im Jahr 1992 wurde M zur Realisierung mehrerer Projekte eingesetzt. Am RIVM selbst wurden das integrierte Modell *TARGETS* (Tool to Assess Regional and Global Environmental and health Targets for Sustainability) (Rotmans u. a., 1994) und das *user support system* (USS) des integrierten Modells *IMAGE* (Alcamo u. a. 1998b; Alcamo 1994) mit M entwickelt. Das USS nutzt allerdings hauptsächlich die Funktionalitäten zur Visualisierung der Modellstruktur und der Modellergebnisse für unterschiedliche Szenarien – die Ergebnisse selbst wurden zuvor über ein nicht mit M entwickeltes Modell berechnet. Die Programmiersprache von M wird im USS lediglich für kleinere zusätzliche Berechnungen benutzt. Neben diesen Systemen gibt es weitere, die sowohl innerhalb als auch außerhalb des RIVM mit M entwickelt wurden. Eine aktuelle Übersicht über diverse Projekte ist auf der Internet-Seite des M-Projektes (`http://www.m.rivm.nl`) zu finden.

ECOBAS

ECOBAS (Benz u. a., 2001; Hoch u. a., 1998) ist ein Software-System zur Erstellung und integrierten Dokumentation von Simulationsmodellen und wurde im Bereich der ökologischen Modellierung entworfen. Ziel des Systems ist die engere Verzahnung von Modellerstellung und Dokumentation. Das System erlaubt die Eingabe von Modellgleichungen und überprüft diese gleichzeitig unter Verwendung der anzugebenden Einheiten für die einzelnen Variablen. Zu den Modellvariablen werden ausführliche Informationen wie Akronyme, Einheiten, Bedeutungen, Typ-Informationen, Werte-Bereiche und allgemeine Beschreibungen gespeichert. Basierend auf den Modellgleichungen kann ECOBAS Source-Code verschiedener Formate erzeugen: FORTRAN, SIMPLEX, EXTEND und

[20] Pipes bieten die Möglichkeit der Interprozesskommunikation zwischen Prozessen, die auf ein und demselben Rechner laufen und die gemeinsame Vorfahren haben. Nähere Einzelheiten zum Thema Pipes finden sich z. B. in Gulbins und Obermayr (1995) und Herold (1999a).

SciLab (die Anbindung an andere Programme ist in Planung). Beachtenswert ist an diesem System, neben der ausführlichen Modelldokumentation, die Verwaltung von Referenzen auf Literatur und auf Personen, die an der Modellentwicklung beteiligt sind bzw. waren.[21]

3.2 Standards

3.2.1 Standardisierungs-Organisationen

Bei der Erstellung von interoperablen, offenen Software-Systemen ist die Verwendung von allgemein anerkannten Methoden, Verfahren, Konzepten und Notationen hilfreich. Die formale Vereinheitlichung des allgemein anerkannten – oder anzuerkennenden – erfolgt durch *Normungen*. Otto Kienzle, Mitbegründer des Deutschen Instituts für Normung (DIN), beschrieb Normung als „... die einmalige, bestimmte Lösung einer sich wiederholenden Aufgabe unter den jeweils gegebenen wissenschaftlichen, technischen und wirtschaftlichen Möglichkeiten."[22] Etwas formeller ist die 'genormte' Form der Definition: „Normung ist die planmäßige, durch die interessierten Kreise gemeinschaftlich durchgeführte Vereinheitlichung von materiellen und immateriellen Gegenständen zum Nutzen der Allgemeinheit." (DIN 820 Teil 1) Normung

Normen (Standards[23]) können unterteilt werden in *nationale Normen*, *internationale Normen* und *Fachnormen*. Für die Normierung sind länder- und fachspezifische Normungsorganisationen zuständig. Zu den nationalen Normierungsorganisationen gehören beispielsweise das American National Standards Institute (ANSI) und das bereits erwähnte Deutsche Institut für Normung (DIN). Die International Organization for Standardization (ISO) ist eine internationale Organisation. Das Institute of Electrical and Electronics Engineers (IEEE) ist ein Beispiel für eine fachliche Standardisierungsorganisation. Norm-Arten

Die hinsichtlich der Entwicklung eines SISA wichtigsten Normungsorganisationen werden nachfolgenden kurz vorgestellt. Einige für die Entwicklung eines SISA relevante Normen werden in den folgenden Abschnitten kurz beschrieben.

[21] Einen ähnlichen Ansatz zur Beschreibung von Modellen verfolgt Maxwell (1999) mit seinem 'parsi-model approach'. Ein Modell besteht hier aus einer Modellbeschreibung und dem ablauffähigen Programm. Modelle setzten sich aus Modulen zusammen, die eine deklarative Beschreibung des Verhaltens beinhalten; das dynamische Verhalten übernimmt eine Modellumgebung. Zur Modellbeschreibung wurde die so genannte *modular modeling language* entwickelt.

[22] Quelle: `http://www.din.de/portrait/definiti.html`

[23] Die Begriffe 'Norm' und 'Standard' werden in dieser Arbeit synonym verwendet. Zur Abgrenzung der Begriffe s. z. B. Bartelme (2000).

ISO

Die *International Organization for Standardization (ISO)*[24] ist ein weltweiter Zusammenschluss der nationalen Standardisierungsinstitute von mehr als 140 Ländern. Die ISO wurde 1947 gegründet und ist eine Nicht-Regierungsorganisation.

Ziel

Die Mission der ISO ist die Unterstützung der Entwicklung von Standardisierungen und damit zusammenhängender Aktivitäten in der Welt im Hinblick auf die Unterstützung des internationalen Austauschs von Gütern und Dienstleistungen, und der Entwicklung von Kooperationen im Bereich der geistigen, wissenschaftlichen, technologischen und ökonomischen Aktivitäten. Die Arbeit der ISO resultiert in internationalen Vereinbarungen, die in internationalen Standards veröffentlicht werden.

Organisation

Während die strategischen Entscheidungen von den ISO-Mitgliedern (also den nationalen mit der Standardisierung beauftragten Organisationen) getroffen werden, wird die technische Arbeit der ISO auf fast 3000 technische Ausschüsse, Unterausschüsse und Arbeitsgruppen verteilt. Diese Gremien setzen sich zusammen aus gleichberechtigten Repräsentanten aus Industrie, Verbraucherverbänden, internationalen Organisationen, wissenschaftlichen Institutionen und Regierungsbehörden. Die Hauptverantwortlichkeit für die Administration eines Ausschusses wird von einem der nationalen Standardisierungsorganisationen (z. B. der DIN) übernommen. Die Koordination des Netzwerks und die Veröffentlichung fertiggestellter Standards übernimmt das in Genf ansässige ISO-Zentralbüro.

Entwicklungsphasen

Die Entwicklung eines internationalen Standards verläuft in sechs Schritten (Rehesaar, 1996):[25]

- Vorbereitungsphase (Phase 0)
- Vorschlagsphase (Phase 1)
- Vorbereitungsphase (Phase 2)
- Ausschussphase (Phase 3)
- Genehmigungsphase (Phase 4)
- Veröffentlichungsphase (Phase 5)

Die *Vorbereitungsphase* (preparation stage) ist eine optionale Phase für die erste Vorbereitung eines Standards. (Die Dauer der Phase ist nicht festgelegt.) In der *Vorschlagsphase* (proposal stage) wird ein so genannter ‘new work item proposal’ (NP) an die Mitglieder der Vollversammlung gesendet, die über den Vorschlag annehmend oder ablehnend entscheiden. Sofern der NP angenommen wurde, wird er allen Mitgliedern zur Abstimmung zugesandt. Die Phase endet mit der Genehmigung des Projektes. (Dauer der Phase: etwa neun Monate.)

[24] Startseite im Internet: http://www.iso.ch

[25] Dargestellt sind die vom *ISO/IEC JTC1 Subcommittee Software Engineering* verwendeten Phasen.

Die *Vorbereitungsphase* dient der Entwicklung eines ‘working draft’ (WD). Dem endgültigen WD gehen i. d. R. mehrere Entwurfs-Versionen voraus, die an einen möglichst großen Leserkreis verteilt werden. Der endgültige WD erhält dann, meist im Rahmen einer Vollversammlung, den Status eines so genannten ‘committee draft (CD)’. (Dauer der Phase: 1 bis 3 Jahre.) In der *Ausschussphase* wird der CD zur Diskussion an den zuständigen Unterausschuss verteilt. Sofern der CD entsprechende Unterstützung durch die Mitglieder erhält, wird der Status auf den so genannten ‘draft international standard’ (DIS) erhöht.[26] (Dauer der Phase: ein bis vier Jahre.) Während der *Genehmigungsphase* wird über den DIS brieflich abgestimmt. Zur Akzeptanz ist eine Zwei-Drittel-Mehrheit erforderlich sowie maximal 25% ablehnender Stimmen und eine Wahlbeteiligung von mindestens 50% der Stimmberechtigten (d. h. der nationalen Standardisierungs-Organisationen)[27]. (Dauer der Phase: max. vier Monate zur Abstimmung.) Ein genehmigter Standard wird dann in der letzten Phase, der *Veröffentlichungsphase*, publiziert.

Über dieses Verfahren soll gewährleistet werden, dass die Standards erst nach ausführlichen Diskussionen auf internationaler Ebene und nachfolgender Genehmigung über eine internationale Abstimmung publiziert werden.

Aufgrund der Wichtigkeit von Geoinformationen wurde 1994 das *Technical Committee Geographic information/Geomatics (ISO/TC 211)*[28] gebildet. Das Mandat für diesen Ausschuss ist die Entwicklung eines integrierten Satzes an Standards für geographische Informationen (die Standard-Reihe 19100). TC 211

ISO/TC 211 arbeitet eng mit anderen Standardisierungsorganisationen zusammen; es existieren beispielsweise Kooperationen mit dem *OpenGIS Consortium* (s. u.), dem gemeinsamen technischen Ausschusses der ISO und der International Electrotechnical Commission (*ISO/IEC JTC1*)[29] sowie der gemeinnützigen Organisation *Global Spatial Data Infrastructure*[30], die ihrerseits eng mit den Vereinten Nationen zusammenarbeitet.

Die auf dem Arbeitsprogramm des ISO/TC 211 stehenden Standards reichen von einem allgemeinen Referenzmodell (ISO 19100) bis hin zu Sensor- und Datenmodelle für Bilder und Rasterdaten (ISO 19130) und Implementierungs-Spezifikationen für Metadaten (ISO 19139). Ein Standard dieser Reihe ist auch für die Entwicklung der SISA-Architektur relevant (19119), der daher im Abschnitt 3.2.5 (Seite 55) näher beschrieben wird. Einen Überblick über die der- Standards

[26]Fehlt der Konsens über den CD, kann das JTC1 die Veröffentlichung als so genannten ‘type 1 technical report’ (TR1) erbitten.

[27]Bei Ablehnung kommt auch hier die Überarbeitung des DIS oder die Veröffentlichung als TR1 in Frage.

[28]Startseite im Internet: `http://www.isotc211.org`

[29]Startseite im Internet: `http://www.jtc1.org`. Eine kurze Beschreibung der Struktur, Mitglieder und Produkte des JTC1 findet sich bei Rehesaar (1996).

[30]Startseite im Internet: `http://www.gsdi.org`

zeitigen Standardisierungsthemen liefert die im Anhang befindliche Tabelle C.3 (Seite 240). Der Bezug von ISO-Standards ist i. d. R. kostenpflichtig.

IEEE

Das Institute of Electrical and Electronic Engineers (IEEE)[31] ist ein gemeinnütziger, technischer Berufsverband mit mehr als 377000 Mitgliedern aus über 150 Ländern. Ziel des IEEE ist die Unterstützung der Entwicklung, Integration, gemeinsamen Nutzung und Anwendung elektro- und informationstechnischen Wissens. Eine Aktivität des IEEE ist die *IEEE Standards Assosiation (IEEE-SA)*, die zur Formulierung und Förderung der international anerkannten IEEE-Standards bevollmächtigt ist. Mitglieder im IEEE-SA sind sowohl Einzelpersonen als auch Firmen.

Ziel

Standards

Von den fast 900 verabschiedeten Standards kann einer beim Aufbau eines SISA besonders relevant werden: der *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) (IEEE 1516)*. Dieser Standard wurde als eine Referenz-Architektur für die verteilte Modellierung und Simulation entwickelt. Abschnitt 3.2.2 (Seite 47) widmet sich diesem Standard. Der Bezug von IEEE-Standards ist, ebenso wie der von ISO-Standards, i. d. R. kostenpflichtig.

W3C

Das *World Wide Web Consortium (W3C)*[32] ist die Organisation, in der die Kerntechnologien des World Wide Web (WWW) entwickelt werden. Das W3C wurde 1994 am Labratory for Computer Science des Massachusetts Institut of Technology (MIT) von Tim Berners-Lee (dem ‘Erfinder’ des WWW) in Kooperation mit seinem ehemaligen Arbeitgeber und Ursprungsort des WWW, dem Europäisches Kernforschungszentrum CERN (Conseil Européen pour la Recherche Nucléaire), gegründet.

Ziel

Das W3C hat sich zur Aufgabe gesetzt, die technische Weiterentwicklung des WWW durch die Förderung von Interoperabilität und offenen Diskussionsforen zu leiten. Auf langfristige Sicht strebt das W3C ein WWW an, auf das alle Menschen unter Berücksichtigung der vielen Unterschiede in Sprache, Ausbildungsstand, materiellen Ressourcen usw. zugreifen können *(universal access)*, das jedem Benutzer die bestmögliche Nutzung der verfügbaren Web-Ressourcen erlaubt *(semantic web)* und dessen Entwicklung die vielfältigen rechtlichen, kommerziellen und sozialen Fragen berücksichtigt, die durch die Technologie aufkommen (*web of trust*). Das W3C sieht seine Rolle darin, Visionen des zukünftigen WWW zu erstellen, Web-Technologien zu deren Realisierung zu entwickeln und sich an Standardisierungsbemühungen zu beteiligen.

[31] Startseite im Internet: `http://www.ieee.org`

[32] Startseite im Internet: `http://www.w3.org/Consortium`

Organisation

Das W3C besteht aus dem so genannten W3C-Team und derzeit etwa 450 weiteren Mitgliedern und ist in Gruppen organisiert: Die technischen Entwicklungen liegen in der Hand von *Working Groups*, während *Interest Groups* für allgemeinere Arbeiten zuständig sind und die Verbindung miteinander in fachlicher Beziehung stehender Gruppen Aufgabe der Coordination Groups ist. Die einzelnen Gruppen sorgen auch für die Koordination ihrer Arbeit mit anderen Standardisierungsorganisationen.

Die Resultate des W3C sind technische Berichte, Open Source Software und Dienstleistungen. Zu den technischen Berichten gehören auch die technischen Spezifikationen, die so genannten *Recommendations*. Entwickler Web-basierter Anwendungen kommen um diese Spezifikationen nicht herum – sie sind die Bausteine des WWW.

Standards

Zu den wichtigsten der zurzeit etwa 40 Empfehlungen zählen diejenigen zur *hyper-text markup language (HTML)*, zum *uniform resource locator (URL)* und zum *hyper-text transport protocol (HTTP)*. Der URL-Standard legt den Aufbau von Web-Adressen fest, der HTTP-Standard definiert das Protokoll, mit dem Web-Seiten von einem Rechner zum anderen übertragen werden und der HTML-Standard definiert die Beschreibungssprache für Web-Seiten und erlaubt damit deren Aufbau (inhaltlich und darstellerisch).

Neben diesen Standards sind für die Realisierung des SISA weitere Spezifikationen des W3C relevant: der *uniform resource identifier (URI)*, die *extensible markup language (XML)* und die mit der XML in Verbindung stehenden Standards zum *XML Schema*, *resource description framework (RDF)* und *document object model (DOM)*. Nähere Informationen zu diesen Empfehlungen finden sich im weiteren Verlauf der Arbeit. Die Empfehlungen des W3C sind kostenfrei und online verfügbar (http://www.w3.org).

Verfügbarkeit

OGC

Das *Open GIS Consortium (OGC)*[33] ist ein internationales Industrie-Konsortium von mehr als 230 Firmen, Regierungsorganisationen und Forschungseinrichtungen. Ziel dieses 1994 gegründeten Zusammenschlusses ist die Entwicklung frei verfügbarer Spezifikationen im Bereich der Geodatenverarbeitung. Die offenen Schnittstellen und Protokolle, die in so genannten *OpenGIS Specifications* definiert werden, unterstützen interoperable Lösungen; die Informationstechnologie soll 'geofähig' gemacht werden und Entwickler befähigen komplexe räumliche Informationen und Dienste für alle denkbaren Anwendungen zugreifbar zu machen.

Spezifikationen

Das OGC unterscheidet zwischen *OpenGIS Abstract Specifications* und *OpenGIS Implementation Specifications*. Erstere spezifizieren die konzeptionelle Basis für Entwicklungsaktivitäten und stellen ein Referenzmodell für die

Abstract vs. Implementation

[33] Startseite im Internet: http://www.opengis.org

Entwicklung der Implementation Specifications bereit. Die OpenGIS Implementation Specifications sind technische Spezifikationen, die Teile der Abstract Specification für spezielle Plattformen zur Entwicklung verteilter Anwendungen (z. B. OLE/COM oder CORBA) spezifizieren. Hier finden sich Programmierrichtlinien für Software-Entwickler zur Integration und Nutzung der OGC-Schnittstellen und OGC-Protokolle.

Abstract Specifications

Die Abstract Specifications sind in Themengebiete (topics) eingeteilt. Derzeit existieren 16 themenzentrierte Abstract Specifications, die jeweils bestimmte Funktions- oder Technologiebereiche betreffen: angefangen von der Definition geographischer Merkmale über die Beschreibung geographischer Referenzsysteme bis hin zu Diensten für die Koordinaten-Transformation von Bildern. Eine Auflistung der OpenGIS Abstract Specifications findet sich im Anhang (Tab. C.4, Seite 241). Auf die in der Liste zu findenden Spezifikationen der *Catalog Services* sowie die Spezifikationen von Metadaten (die das OGC im Mai 2001 von der ISO (ISO/DIS 19115) übernahm) wird im Abschnitt 5.2.1 (Seite 96) näher eingegangen. Eine Übersicht über die Abhängigkeiten der OGC-Spezifikationen untereinander findet sich ebenfalls im Anhang (Abb. C.1, Seite 237).

Verbreitung

Die Akzeptanz und Anwendbarkeit der OGC-Spezifikationen zeigt sich sowohl im wissenschaftlichen Bereich – im Rahmen der Umweltinformatik z. B. bei Voges (2001), Senkler (2001) und Fitzke und Müller (2000) – als auch im kommerziellen Bereich – z. B. durch die Umsetzung von OGC-Spezifikationen[34] innerhalb des Produkts ArcGIS der Firma ESRI[35].

Weitere Organisationen

Neben den angeführten Organisationen gibt es weitere, wie die *Simulation Interoperability Standards Organization (SISO)*[36] und die *Object Management Group (OMG)*[37], deren Arbeiten bei der Entwicklung eines SISA ebenfalls zu Rate gezogen werden können. Einige internationale Normen werden vom *Deutschen Institut für Normung (DIN)*[38] in die deutsche Sprache übersetzt und in die eigene Normenreihe aufgenommen.

IETF

Sofern Internet-Technologien für die Realisierung eines SISA eingesetzt werden sollten, sind die Veröffentlichungen der *Internet Engineering Task Force (IETF)* von besonderer Bedeutung. Die IETF ist ein internationaler und offener Verbund aus Netzwerk-Entwicklern, -Betreibern, -Anwendern und Wissenschaftlern, die sich mit der Entwicklung der Internet-Architektur und deren

[34] *OpenGIS Simple Features Specification for OLE/COM 1.1* und *OpenGIS Simple Features Specification for SQL 1.1.*

[35] Startseite im Internet: `http://www.esri.com`

[36] Startseite im Internet: `http://www.sisostds.org`

[37] Startseite im Internet. `http://www.omg.org`

[38] Startseite im Internet: `http://www.din.de`

Betrieb beschäftigen. Alle Spezifikationen des IETF werden im Internet als so genannte *Requests for Comments (RFCs)* veröffentlicht.[39] Zu den wichtigen Standards gehört z. B. der RFC 791, der das Internet-Protokoll spezifiziert. Im Rahmen der Realisierung der SISA-Architektur (Kapitel 5, Seite 95) bekommen weitere RFCs eine besondere Bedeutung. RFC

Weitere Informationen zum Thema Interoperabilität und Normung finden sich bei Bartelme (2000). Eine Übersicht über Standardisierungs-Organisationen im Bereich der Geoinformation gibt Carson (2000).

3.2.2 High Level Architecture (HLA)

Die *High Level Architecture (HLA)* (Kuhl u. a., 1999) ist eine Architektur zur Verbindung von interagierenden Teilmodellen zu Gesamtmodellen und verfolgt das Ziel, die Interoperabilität von Simulationsmodellen zu erhöhen. Entwickelt im militärischen Umfeld[40], hält die Architektur zunehmend im zivilen Bereich Einzug (s. z. B. Schulze u. a., 1999) und wurde im Jahr 2000 zum IEEE-Standard (IEEE, 2000a)[41].

Die grundlegenden Elemente der HLA sind die so genannten *Federates* und *Federations*: ein Federate ist eine Anwendung, die sich an einer Federation beteiligt und kann mit einem Teilmodell verglichen werden[42]. Eine Federation besteht – vergleichbar mit einem Gesamtmodell – aus einer Menge interagierender Federates, einer formalen Beschreibung des gemeinsamen Objektmodells und einer Infrastruktur, die für die Kommunikation zwischen den Federates zuständig ist (vgl. Abb. 3.6, Seite 50).

Die formale Definition der *Modeling and Simulation High Level Architecture* – so der offizielle Name der Architektur – umfasst drei Hauptkomponenten:

- HLA Rules (IEEE, 2000a)
- HLA Object Model Template (OMT) (IEEE, 2000c)
- HLA Federate Interface Specification (IEEE, 2000b)

Die *HLA Rules* beschreiben die Hauptbestandteile einer Federation und definieren über insgesamt zehn Regeln das Zusammenspiel zwischen Federate und Federation. Hier wird beispielsweise festgelegt, dass sich die Federates auf ein gemeinsames Objektmodell – das *Federation Objekt Model (FOM)* – beziehen müssen und dass jeglicher Datenaustausch zwischen den Federates über eine Rules

[39] Die Spezifikation der Standardisierungsprozesses selbst ist ebenfalls ein RFC und findet sich unter `http://www.ietf.org/rfc/rfc2026.txt?number=2026`.

[40] Die HLA ist seit 1996 die Standard-Architektur für Simulationsanwendungen im Department of Defence der USA. Siehe `http://www.dod.mil`.

[41] Grundlage für die Beschreibung der HLA in diesem Unterabschnitt liefert die *HLA Technical Specification, Version 1.3*, die die Vorlage für den IEEE-Standard 1516 darstellt u. über `https://www.dmso.mil/public/transition/hla/techspecs` erhältlich ist.

[42] Ein Federate kann aber auch jedes andere beteiligte Programm sein, wie z. B. ein Sensor zur Datenerfassung oder eine passive Anwendung zur Datenvisualisierung.

Instanz namens *Run-Time Infrastructure (RTI)* ablaufen muss. Weiterhin wird in den Regeln festgelegt, dass es zu jedem Federate eine formale Beschreibung seiner Objekte in Form eines *Simulation Object Model (SOM)* geben muss. Die einzelnen Regeln sind in Anhang C.2 (Seite 243) dokumentiert.

OMT

Im *Object Model Template (OMT)* werden die Formate und die Syntax definiert, die zur formalen Definition von Objekten, Attributen, Interaktionen und Parametern – also zur Erstellung von FOMs und SOMs – benutzt werden müssen.

Interface Specification

Die *HLA Federate Interface Specification* definiert die Schnittstelle zwischen den Federates und der *Run-Time Infrastrukture (RTI)*. Die Dienste der RTI werden eingeteilt in sechs Gruppen:

1. Federation-Management (federation management)
2. Deklarations-Management (declaration management)
3. Objekt-Management (object management)
4. Eigentum-Management (ownership management)
5. Zeit-Management (time management)
6. Datenverteilungs-Management (data distribution management)

Federation

Die Gruppe für das *Federation-Management* umfasst Dienste zur Erzeugung und zum Löschen so genannter *Federate Executions* (Objekte, die eine Federation zur Laufzeit repräsentieren) sowie zur dynamischen An- und Abmeldung, zur Sychronisationssteuerung und zum Speichern und Wiederherstellen von Federates. Bevor ein Federate in eine *Federation Execution* eintreten kann, muss die *Federation Execution* existieren.

Deklaration

Die Dienste des *Deklarations-Managements* bieten die Möglichkeit Objekt-Klassen und Interaktions-Klassen zu veröffentlichen und Klassen oder einzelne Attribute zu 'abonnieren' – die HLA funktioniert nach dem *Publisher-Subsciber-Prinzip*: Objekte, die die Werte ihrer Attribute anderen Objekten mitteilen wollen, 'veröffentlichen' diese Attribute, während Objekte, die Interesse an diesen Attribut-Werten haben, die Attribute 'abonnieren' können. Die Änderung eines Attributwertes wird dann automatisch (über die RTI) allen interessierten Objekten bekannt gegeben[43].

Objekte

Für das *Objekt-Management* werden Dienste bereitgestellt zur Registrierung von Objekt-Instanzen bei einer Federation, zur Aktualisierung von Attribut-Werten und Versendung von Interaktionen sowie zur Steuerung des Transportes von Attribut-Werten und Interaktionen. Über die Dienste des *Eigentum-Managements*[44] können die Eigentumsrechte einzelner Objekt-Attribute verwaltet werden.

Eigentum

Es gibt Dienste zur Abfrage der Eigentumsverhältnisse, zur

[43] Zum 'Publisher-Subscriber-Muster' s. z. B. Gamma u. a. (1996) oder Buschmann u. a. (1998).

[44] Der Eigentümer eines Objekt-Attributes hat mehr Möglichkeiten bei der Verwaltung des Attributes und das Recht Attributwerte zu ändern. Die Verwaltung der Rechte geschieht in der HLA dynamisch.

Weitergabe des Eigentumsrechtes (inklusive der Anfrage das Eigentum zu erlangen). Die *Zeit* des modellierten Systems wird in der Federation als Punkt auf einer Federation-Zeitachse repräsentiert. Die Zeit innerhalb der Federates läuft entweder eingeschränkt oder uneingeschränkt entlang dieser Zeitachse ab. Die Dienste des *Zeit-Managements* bieten einen Mechanismus zur Kontrolle des Zeitverlaufes jedes Federates entlang der Federation-Zeitachse. Federates können *time regulated* oder *time constrained* sein: im ersteren Fall assoziiert ein Federate über 'Zeitstempel' Aktivitäten mit Punkten auf der Federation-Zeitachse, im zweiten Fall ist ein Federate interessiert am Empfang von Benachrichtigungen über solche Aktivitäten.[45] Während das Deklarations-Management den Datenaustausch auf der Ebene von Klassen-Attributen regelt, stellt das *Datenverteilungs-Management* Dienste zur Verfügung, die von den Federates benutzt werden können, um den Datenaustausch zu reduzieren. Die Dienste erlauben es, so genannte *nutzerdefinierte Räume (user defined spaces)* zu definieren. Die RTI gibt Daten und Interaktionsanfragen dann nur an diejenigen Federates weiter, die sich für diesen Ausschnitt interessieren.

Zeit

Datenverteilung

Die Einteilung in die sechs Gruppen sollte es Entwicklern ermöglichen, nur die für sie relevanten Dienste zu implementieren und die anderen ignorieren zu können (Kuhl u. a., 1999).

Die grundlegenden statischen und dynamischen Prinzipien der HLA werden in den Abbildungen 3.6 (Seite 50) und 3.7 (Seite 51) noch einmal zusammenfassend dargestellt.

Entwicklungsprozess

Neben den drei Standards zur Beschreibung des allgemeinen Rahmens (IEEE, 2000a), der Schnittstellen (IEEE, 2000b) und der Modell-Beschreibungssprache (IEEE, 2000c) enthält die Standard-Serie IEEE 1516 noch ein Dokument, das einen Entwicklungsprozess beinhaltet, der für die Erstellung HLA-konformer Simulationsmodelle empfohlen wird (IEEE, 2003).

Die HLA ist eine *Architektur* und stellt damit lediglich die Prinzipien bereit, die zur Erhöhung der Interoperabilität von Simulationsmodellen beiträgt. Um diese Prinzipien einzusetzen, ist es notwendig, die *RTI* zu *implementieren.* Bis September 2002 stellte das *Defense Modeling and Simulation Office (DMSO)* der USA[46] eine Implementierung kostenfrei zur Verfügung. Seither wird auf kommerzielle Implementierungen der RTI verwiesen. Eine Liste aktueller RTI-Software – die derzeit für die Programmiersprachen C++ und JAVA verfügbar ist – ist auf den Internet-Seiten des DMSO zu finden[47].

[45] Standardmäßig sind Federates weder *time regulated* noch *time constrained.* Sofern keine Änderungen vorgenommen werden, machen die Federates keinen Gebrauch von den Zeit-Management-Diensten.

[46] Startseite im Internet: `https://www.dmso.mil`

[47] `https://www.dmso.mil/public/transition/hla/rti/statusboard`

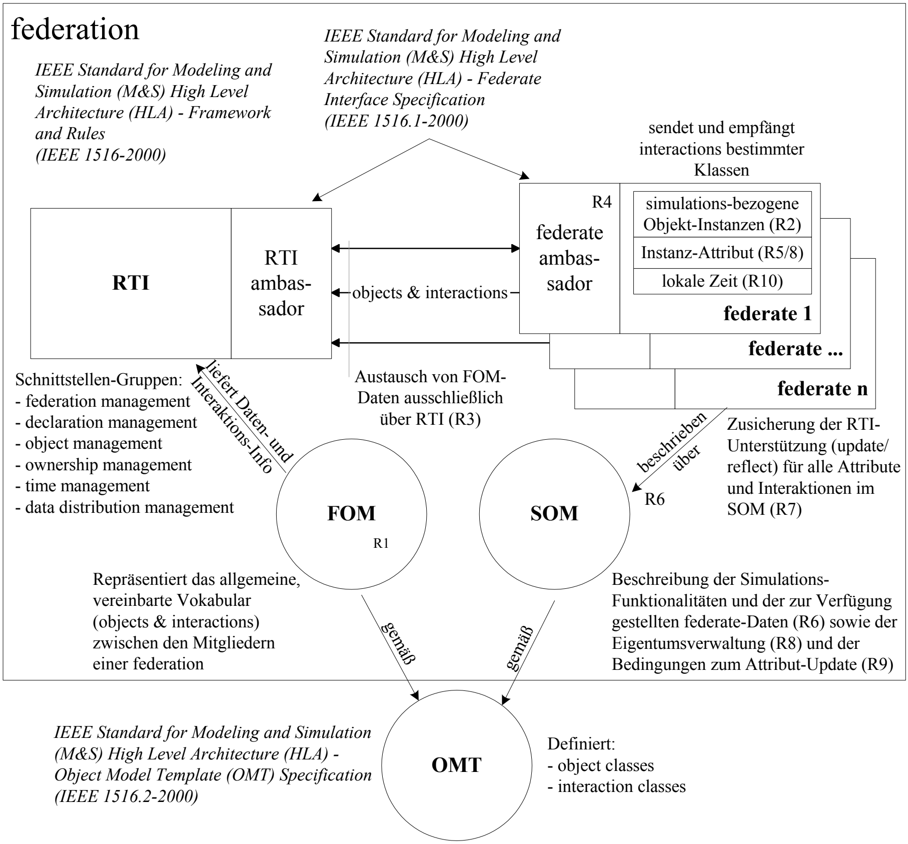


Abbildung 3.6: High Level Architekture – Übersicht. Eine *federation* (das ‘Gesamtmodell’) besteht aus mehreren *federates* (Teilmodellen) und der *run-time infrastructure (RTI)*, über die der gesamte Datenaustausch abgewickelt wird. Das Gesamtmodell wird über das *federation object model (FOM)* beschrieben; die dem Gesamtmodell zur Verfügung gestellten Daten und Funktionen der einzelnen Teilmodelle werden im *simulation object model (SOM)* definiert. Die formale Spezifikation der *High Level Architecture* besteht aus drei Teilen: IEEE 1516 beschreibt den allgemeinen Rahmen und die zehn Regeln (R1-R10), die das Zusammenwirken der Federates (Teilmodelle) innerhalb einer Federation (Gesamtmodell) definieren; IEEE 1516.1 spezifiziert die Schnittstelle zwischen der RTI und den einzelnen Teilmodellen; IEEE 1516.2 definiert die Beschreibungssprache, die für das FOM und das SOM benutzt werden muss. Die Regeln finden sich im Anhang C.2 (Seite 243). Weitere Informationen finden sich im Text.

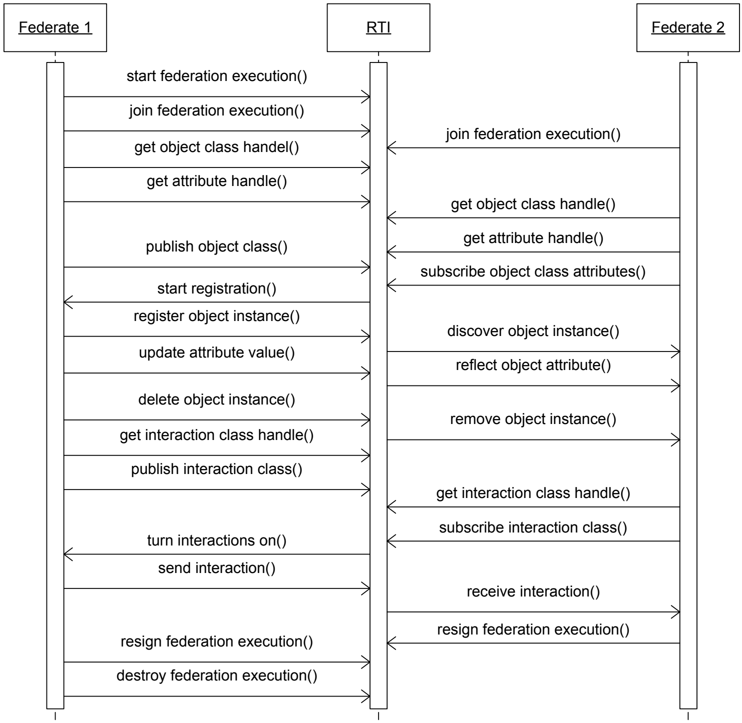


Abbildung 3.7: HLA-Prinzip der Daten- und Interaktionsweitergabe. Die Instantiierung einer Federation wird durch den Aufruf *start federation execution()* bei der RTI veranlasst. Im Anschluss daran können sich Federates dieser Federation durch Aufruf von *join federation execution()* anschließen – im abgebildeten Beispiel sind es zwei Federates (Federate1 und Federate2). Bevor Federates Daten über Objekte und deren Attribute austauschen können, müssen sie sich zunächst über *get object/attribute handle()* geeignete Identifizierungsnummern besorgen. Federate1 kann nun die 'Veröffentlichung' von Objekten anmelden. Diese Anmeldung geschieht über *publish object class()*. Ist Federate2 an derartig veröffentlichten Attributen dieses Objektes interessiert, kann es dies über *subscribe object class attribute()* der RTI bekannt geben. Ab dann erfolgt eine automatische Benachrichtigung von Federate2, sobald neue Objekte von Federate1 registriert werden oder sich die Attributwerte registrierter Objekte ändern (*update/reflect object()*). Die Registrierung von Interaktionen findet ebenfalls nach dem Muster *publish/subscribe* statt. Eine Federation existiert so lange, bis sich alle Federates über *resign federation execution()* abgemeldet haben. Die endgültige Zerstörung der Federation geschieht über den Aufruf der Funktion *destroy federation execution()*.

3.2.3 NIST/ECMA-Referenz-Modell

Das NIST/ECMA-Referenz-Modell[48] wurde ursprünglich als Architektur zur Integration verschiedener Systeme (Anwendungen) im Rahmen des Computer Aided Software Engineering (CASE) entwickelt. Neben der Verfolgung des Ziels verschiedene Datenquellen einheitlich zugreifbar zu machen, stellt die Architektur einen erweiterbaren Rahmen bereit, der für die Kommunikation der einzelnen Anwendungen zuständig ist und der eine einheitliche Benutzungsoberfläche und Repräsentation der Daten bietet. Abbildung 3.8 zeigt die Übersicht dieses Modells.

Ziel

Komponenten

Die einzelnen Komponenten besitzen nach Hering u. a. (2000) die folgenden Verantwortlichkeiten: *Repository Services* sind zuständig für die Speicherung der Entwicklungsdaten; die *Data-integration Services* sind für die Versions- und Konfigurationsverwaltung zuständig und erlauben einen transparenten Zugriff auf die Entwicklungsdaten; die *Process-management Services* sind als Abstraktionsebene zu den Werkzeugen (*Tools*) und als Schicht zwischen Benutzungsschnittstelle und den eigentlichen Werkzeugen zuständig für die Verwaltung der Zugriffsmöglichkeiten der Anwender; die *User-interface Services* stellen eine einheitliche Benutzungsschnittstelle für alle Werkzeuge bereit; die *Message Services* erlauben den Informationsaustausch zwischen den einzelnen Diensten und den zwischen den Werkzeugen.

Die Problemstellung, die zur Entwicklung dieser Architektur geführt haben – der konsistenten Integration von Daten und Programmen (vgl. Shaw und Garlan, 1996) – ist vergleichbar mit den Integrationsproblemen, die bei integrierten Modellen auftreteten. Da das NIST/ECMA-Referenz-Modell zur Entwicklung der Referenz-Architektur des Open-GIS-Konsortiums, die in Unterabschnitt 3.2.5 (Seite 55) genauer vorgestellt wird, herangezogen wurde, sei für weitere Ausführungen zum NIST/ECMA-Modell auf Chen und Norman (1992) verwiesen.

3.2.4 Open Distributed Processing – Reference Model

Die Komponenten eines SISA sollten nicht zwangsläufig auf einem Host (Rechner) laufen müssen. Eine Verteilung des Systems ist bei der Architekturentwicklung daher zu berücksichtigen.

Ein verteiltes System stellt besondere Anforderungen an die Software-Architektur.[49] Der gemeinsame technische Ausschuss der ISO und der IEC (JTC1) (siehe Abschnitt 3.2.1, Seite 41) hat daher ein Rahmenwerk geschaffen, das die Erstellung von Standards für solche Systeme erleichtern und fördern soll: den

[48]NIST = National Institute of Standards and Technology, ECMA = European Computer Manufacturers Association.

[49]Zum Beispiel Anforderungen bezüglich der Synchronisation von Komponenten und der Fehlerbehandlung. Zu den Besonderheiten verteilter Systeme s. ISO (1998).

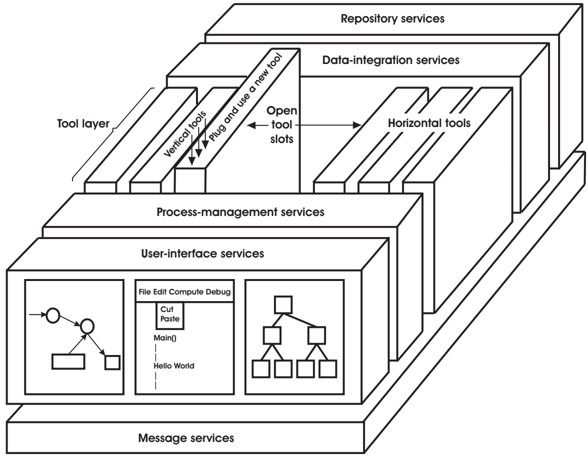


Abbildung 3.8: NIST/ECMA Referenz-Architektur. Erklärungen finden sich im Text. Quelle: Chen und Norman (1992).

Standard *ISO/IEC 10746 Information Technology – Open Distributed Processing – Reference Model* (ISO, 1998). Der Standard, der auch als *ISO RM-ODP* oder kurz *RM-ODP* bezeichnet wird, besteht aus vier Teilen:

ISO/IEC 10746-1 Overview

ISO/IEC 10746-2 Foundations

ISO/IEC 10746-3 Architecture

ISO/IEC 10746-4 Architectural Semantics

Der erste Teil motiviert die Anwendung offener, verteilter Systeme, zeigt deren Möglichkeiten auf, erklärt Schlüsselkonzepte und skizziert die ODP-Architektur. Darüber hinaus enthält dieser Teil Beispiele zur Anwendung des Standards für potentielle Nutzer, d. h. für Autoren von abgeleiteten Standards und für Entwickler von ODP-Systemen. Der zweite Teil enthält die Definitionen der Konzepte und des analytischen Rahmenwerks zur normalisierten Beschreibung beliebiger verteilter Systeme. Der dritte Teil des Standards benutzt die Beschreibungstechniken des zweiten Teils und beschreibt die Eigenschaften, die ein verteiltes System aufweisen muss, um als *offen* zu gelten. Abgeleitete ODP-Standards müssen die hier aufgeführten Kriterien erfüllen. Der vierte Teil beschreibt eine Vereinheitlichung einiger Modellierungskonzepte des zweiten Teils unter Verwendung standardisierter formaler Beschreibungssprachen.

Overview

Foundation

Architecture

Semantics

Zur Standardisierung offener verteilter Systeme werden vier grundlegende Elemente gefordert:

- ein objektorientierter Modellierungsansatz zur Systembeschreibung
- die Systembeschreibung über fünf gesonderte, aber in wechselseitiger Beziehung stehende Sichtweisen (viewpoints)
- die Definition einer System-Infrastruktur zur Verteilungstransparenz für System-Anwendungen
- ein Rahmenwerk zur Feststellung, ob ein System standardkonform ist

Der objektorientierte Ansatz wird wegen der etablierten Entwurfsmethoden der Abstraktion und Kapselung gewählt. Durch die Abstraktion kann die Systemfunktionalität beschrieben werden, ohne dass auf die Details der Implementierung eingegangen werden muss (die Kapselung erlaubt beispielsweise die Heterogenität oder die Implementierung von Sicherheitskonzepten zu verbergen). Als weiterer Pluspunkt der Objektorientierung wird die allgemeine Bekanntheit ihrer Konzepte betrachtet.

fünf Sichtweisen

Im ersten Teil des Standards wird eine Architektur-Beschreibung eingeführt, die auf fünf unterschiedlichen Sichtweisen auf ein System beruht: den *enterprise viewpoint*, den *computational viewpoint*, den *information viewpoint*, den *engineering viewpoint* und den *technology viewpoint*. Eine zusammenfassende Beschreibung der einzelnen Sichtweisen findet sich in Tabelle 3.1. Eine Anwendung des Standards ist die im folgenden Unterabschnitt vorgestellte Dienste-Architektur des OpenGIS-Konsortiums. Für weitere Details sei an dieser Stelle direkt auf den Standard verwiesen (ISO, 1998); eine Einführung in den Standard liefern Farooqui u. a. (1995) und Schürmann (1995).

Sichtweise	Beschreibung
Enterprise Viewpoint	Sichtweise auf ein ODP-System und seine Umgebung, die sich auf den Zweck, den Bereich und die Grundsätze für dieses System konzentriert.
Computational Viewpoint	Sichtweise auf ein ODP-System und seine Umgebung, die durch eine funktionale Zerlegung des Systems in über Schnittstellen interagierende Objekte eine Verteilung erlaubt.
Information Viewpoint	Sichtweise auf ein ODP-System und seine Umgebung, die sich auf die Semantik der Informationen und der Informationsverarbeitung konzentriert.
Engineering Viewpoint	Sichtweise auf ein ODP-System und seine Umgebung, die sich auf die Mechanismen und Funktionen konzentriert, die zur Unterstützung für verteilte Interaktionen zwischen den Objekten des Systems benötigt werden.
Technology Viewpoint	Sichtweise auf ein ODP-System und seine Umgebung, die sich auf die Auswahl von Technologien innerhalb des Systems konzentriert.

Tabelle 3.1: Sichtweise auf ein Software-System nach ISO/RM-ODP (ISO, 1998).

3.2.5 OpenGIS Service Architecture

Das *Open GIS Consortium* schlägt in seinen *Abstract Specifications* (s. S. 45) die *OpenGIS Service Architecture* (Percivall, 2002) vor. Diese Architektur wird als eine von vielen möglichen technischen Referenzmodellen angesehen. Der Architektur liegen die Annahmen zugrunde, dass die Zielsysteme 1) verteilte Systeme und 2) objektorientiert aufgebaut sind.

Bis zur Version 4 definierte die OpenGIS Service Architecture im Sinne der ISO RM-ODP (s. Unterabschnitt 3.2.4) die wesentlichen Teile der *computational view* eines Informationssystems für georäumliche Verarbeitungen; die Architektur stellte also lediglich einen Rahmen (engl. framework) der *Dienste* bereit, die für die Entwicklung und Ausführung georäumlich orientierter Anwendungen benötigt werden.

Im April 2001 wurde über das *OGC Technical Committee* die Übernahme der Arbeiten zum *ISO-Standard 19119 (Geographic information – Services)* als OGC Service Architecture beschlossen. Die derzeit verfügbare Version 4.3 der OGC Service Architecture (Percivall, 2002)[50] beinhaltet den Entwurf zum Internationalen Standard (ISO/DIS 19119[51]).

ISO 19119 liefert eine Taxonomie für geographische Dienste und schreibt vor, wie plattformneutrale Spezifikationen für Dienste zu erzeugen und hierzu konforme plattformspezifische Spezifikationen abzuleiten sind. Inhalt

Die durch den Architektur-Standard für geographische Dienste verfolgten Ziele sind: Ziele

- ein abstraktes Rahmenwerk bereitzustellen, das eine abgestimmte Entwicklung spezifischer Dienste erlaubt
- durch eine Schnittstellen-Standardisierung interoperable Daten-Dienste zu ermöglichen
- durch die Definition von Metadaten über Dienste die Entwicklung von Dienste-Katalogen zu unterstützen
- die Trennung einzelner Daten und Dienste zu ermöglichen
- die Nutzung der Dienste eines Anbieters auf den Daten eines anderen Anbieters zu ermöglichen
- ein abstraktes Rahmenwerk zu definieren, das unterschiedlich implementiert werden kann

Zur Erreichung dieser Ziele erweitert der Standard das architektonische Referenzmodell, das in ISO 19101 definiert ist (ISO 19101 definiert das so genannte *Extended Open Systems Environment (EOSE) model for geographic services*).

[50] OpenGIS AS Topic 12. Alle Verweise auf die ISO 19119 in der vorliegenden Arbeit beziehen sich auf diese Veröffentlichung.

[51] DIS heißt *Draft International Standard* und bezieht sich auf den Status des Dokuments. Ein DIS wird den Mitgliedern der ISO zur Abstimmung vorgelegt um zu einem *International Standard* zu werden.

View-points

ISO/DIS 19119 betrachtet die Service-Architektur aus vier der insgesamt fünf Blickrichtungen der RM-ODP und beschreibt den *computational viewpoint*, den *information viewpoint*, den *engineering viewpoint* und den *technological viewpoint*. Der *enterprise viewpoint* wird in anderen Teilen der ISO-19100-Serie beschrieben (z. B. im Referenzmodell ISO 19101). Die in ISO/DIS 19119 behandelten Sichtweisen werden im Folgenden kurz beschrieben.

Computational Viewpoint

Computational Viewpoint

Bei der Beschreibung des *computational viewpoint* wird die Basis für die Verkettung von Diensten gelegt. Dieser Abschnitt des Standards

- definiert die Konzepte der Dienste (Services), Schnittstellen und Operationen sowie deren Beziehungen untereinander
- definiert ein Modell für die Verkettung von Diensten, um größere Aufgaben lösen zu können (service chaining)
- definiert ein Metadaten-Modell für Dienste, um das Auffinden von Diensten über Kataloge zu unterstützen
- stellt einen Ansatz für die physikalische Verteilung von Diensten (durch die Nutzung einer n-Tier-Architektur) vor

Die einzelnen Punkte werden im Folgenden kurz erläutert.

Dienste-Konzept

Schnittstelle

Das zentrale Element des Dienste-Konzepts (s. Abb. 3.9) ist die Schnittstelle: Eine *Schnittstelle* wird definiert als benannte Menge von Operationen, die das Verhalten einer Entität charakterisiert. Schnittstellen sind abstrakte, von der späteren Realisierungsplattform unabhängige Spezifikationen und werden über *Operationen* definiert. Eine Operation ist die abstrakte Beschreibungen einer über die Schnittstelle angebotenen Aktion zur Datentransformation oder Datenabfrage.

Operation

Port

Damit Schnittstellen von Software-Agenten oder Personen (den Nutzern) benutzt werden können, müssen sie, unter Berücksichtigung plattformabhängiger Spezifikationen, implementiert werden. Eine solche Implementierung wird als *Port* bezeichnet.

Dienst

Am Ende dieser Definitionskette steht der Dienst, der den eigentlichen Wert für den Nutzer darstellt. Der Dienst besteht aus mehreren Ports, d. h. aus den Implementierungen mehrerer Schnittstellen. Ein Dienst, der auf einem bestimmten Rechner läuft und der über ein Netzwerk zugreifbar ist, wird als *Instanz eines Dienstes* bezeichnet. Die Implementierung eines Dienstes kann mit einem für diesen Dienst spezifischen Datensatz verbunden sein oder mit mehreren, unspezifischen Datensätzen arbeiten. Im ersten Fall spricht man von

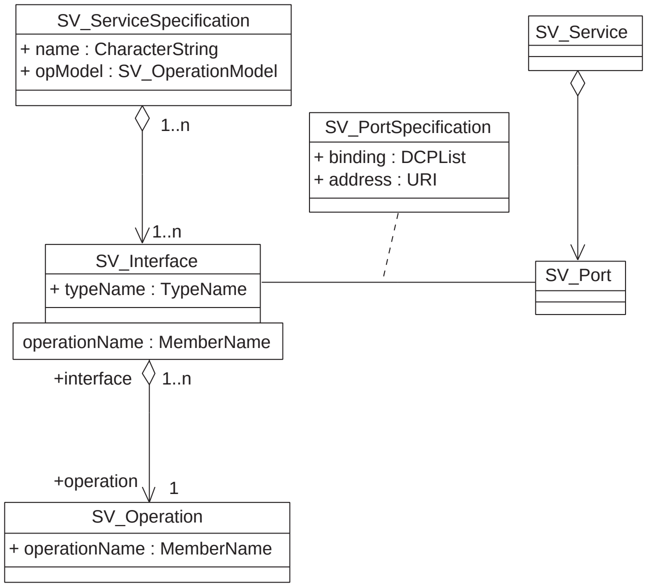


Abbildung 3.9: Statisches Modell des Dienste-Konzepts der ISO/DIS 19119 in UML-Notation. Dienste (Services) basieren auf der plattformabhängigen Implementierung (Ports) von Schnittstellen (Interfaces). Die Schnittstellen werden in einer Dienste-Spezifikation (Service Specification) zusammengefasst und bestehen ihrerseits aus Operationen (Operations). Quelle: ISO/DIS 19119 (Percivall, 2002).

einem *eng gekoppelten Dienst (tightly-coupled service)*, im zweiten Fall von einem *lose gekoppelten Dienst (loosely-coupled service)*. Welche Schnittstellen ein Dienst implementiert, wird in der *Dienst-Spezifikation* festgelegt.

Abbildung 3.9 fasst die Beziehungen der Konzepte in graphischer Form zusammen.

Dienste-Verkettung

Um auch größere Aufgaben über Dienste lösen zu können, wird das Modell der so genannten *Dienste-Ketten (service chains)* definiert. Eine Dienste-Kette wird als gerichteter Graph aufgefasst und definiert als eine Folge von Diensten, in der bei allen miteinander verbundenen Diensten die Durchführung der ersten Aktion für die Durchführung der zweiten Aktion notwendig ist. Dienste-Kette

ISO/DIS 19119 definiert drei Muster für Dienste-Ketten: nutzerdefinierte Verkettung, Workflow-verwaltete Verkettung und aggregierte Dienste. Bei der Muster

nutzerdefinierten Verkettung wird die Dienste-Kette vom Nutzer definiert und kontrolliert. Die *Workflow-Variante* setzt voraus, dass die Kette vordefiniert ist; der Nutzer muss lediglich die Kenntnis von der Kette als Ganzes haben (die Steuerung der einzelnen Dienste übernimmt ein Workflow Service). Bei einem *aggregierten Dienst* tritt die Kette als ein einzelner Dienst auf, der die Koordination der einzelnen Dienste übernimmt. Im Gegensatz zum Workflow-Muster werden die einzelnen Aktionen des aggregierten Dienstes vor dem Nutzer versteckt – ein aggregierter Dienst wird daher auch als 'undurchsichtige Verkettung' (opaque-chaining) bezeichnet. Aggregierte Dienste erlauben eine rekursive Zusammensetzung von Diensten: Eine Dienste-Kette kann zu einem Dienst werden, was die Skalierbarkeit eines Systems fördert.

Dienste-Organisation

Ein System stellt oft viele unterschiedliche Dienste bereitstellt. Da für die Erledigung einer bestimmten Aufgabe aus dieser Gesamtmenge oft nur wenige Dienste anwendbar sind, schlägt ISO/DIS 19119 die Einrichtung so genannter *services organizer folder (SOF)* vor. Ein SOF ist eine Datenstruktur, die Referenzen auf Dienste enthält, die in bestimmten Situationen anwendbar sind. Nutzer eines Systems können ein aufgabenbezogenes SOF erstellen und anderen Nutzern, für die Suche nach Diensten in vergleichbaren Situationen, zur Verfügung stellen.

SOF

Metadaten-Modell für Dienste

ISO/DIS 19119 stellt ein Metadaten-Modell für Dienste (genauer: für die Instanzen von Diensten) bereit. Die über die Metadaten-Elemente bereitgestellten Informationen sollen ausreichen, damit ein Nutzer den Dienst verwenden kann.

Zur Definition des Metadaten-Schemas werden Elemente der ISO 19115 (Geographic information – Metadata) verwendet. Die Metadaten zu einer Dienst-Instanz bestehen aus einer allgemeinen Beschreibung der Funktionalität des *Dienstes* und aus den Beschreibungen der *Operationen*, die durch die Dienst-Instanz aufgerufen werden können. Sofern es sich um eine eng gekoppelte Dienst-Instanz handelt, sind weitere Metadaten gemäß ISO/DIS 19115 für die *Daten* anzugeben.

Dienste-Architektur (simple service architecture)

Für die Implementierung einer nachrichtenbasierten Architektur zur Verkettung von Diensten wird die so genannte *'einfache Dienste-Architektur' (simple service architecture)* vorgeschlagen. Damit sich ein System als Instanz dieser Architektur bezeichnen darf, müssen bei der Implementierung fünf Punkte berücksichtigt werden, die bei Percivall (2002) mit den folgenden, unten näher erklärten Stichpunkten zusammengefasst werden:

- Nachrichten-Operationen
- Trennung von Kontrolle und Daten

- Zustandsbehaftete vs. zustandslose Dienste
- Bekannter Dienst-Typ
- Zugewiesene Hardware (adequate hardware)

Operationen sollten aus Vereinfachungsgründen über Nachrichten modelliert werden. Eine *Nachrichten-Operation ('message operation')* sollte aus einer *Anfrage* und einer *Antwort* bestehen. Anfragen und Antworten führen *Parameter* mit sich, die unabhängig vom Kontext in einer einheitlichen Weise übertragen werden. Bei den Anfrage-Antwort-Interaktionen einfacher Anwendungen gibt es charakteristische *Nachrichtenaustausch-Muster*, wie den unidirektionalen (Ereignis) und den bidirektionalen Austausch. Eine Dienstspezifikation sollte die Erzeugung und Nutzung von Anwendungen so einfach wie möglich machen. Nachrichten-Operation

Trennung von Kontrolle und Daten. Ein Client, der auf einen Dienst zugreift (ihn kontrolliert), benötigt eventuell nicht die gesamten Resultate eines Dienstes; im Falle einer Dienste-Kette benötigt er beispielsweise nur das Endergebnis, nicht aber die gegebenenfalls zur Verfügung stehenden Teilergebnisse. Aus diesem Grund sollten die Operationen einer Schnittstelle die *Kontrolle* eines Dienstes vom Zugriff auf die *Ergebnisdaten* des Dienstes *trennen*. Ein Client sollte die Option haben, nur den Status einer Operation zu empfangen und auf die Daten separat über eine andere Operation zuzugreifen. Kontrolle & Daten

Aus Vereinfachungsgründen sollte ein Dienst zustandslos sein, d. h. der Aufruf eines Dienstes besteht aus *einer* Aufruf-Antwort-Sequenz und es gibt keine Abhängigkeit von vorhergehenden und zukünftigen Interaktionen. Dies ist nicht immer möglich: für einige Dienste müssen Vorbedingungen gesetzt werden, für andere sind Interaktionen notwendig. In solchen Fälle, sind die möglichen Zustände des Dienstes mit Hilfe eines Zustandsdiagramms zu modellieren. Die Zustandsübergänge werden dann durch Operationen ausgelöst. Zustände

Alle Dienst-Instanzen sind von spezifischen Dienst-Typen und der Client kennt den Typ vor der Laufzeit. Clients sollen Software enthalten, die den Dienst-Typ feststellt, bevor sie in Verbindung mit einer Dienst-Instanz treten. bekannter Dienst

Die in ISO/DIS 19119 beschriebenen Dienste sind Software-Implementierungen, die auf Hardware-Hosts laufen. Der Standard nimmt an, dass die mit dem Hardware-Hosting verbunden Software-Fragen transparent gegenüber dem Nutzer sind – in anderen Worten: um die Zuweisung eines Hardware-Hosts zu einem Dienst muss sich der Nutzer nicht kümmern. zugewiesene Hardware

Information Viewpoint

Ziel der Festelegungen zum *information viewpoint* ist die Interoperabilität[52] des Informationsmodells. Zwei Systeme werden nach ISO 19119 dann als interoperabel angesehen, wenn sie sowohl *syntaktisch* als auch *semantisch* interoperabel

[52] Zum Begriff der Interoperabilität siehe Unterabschnitt 2.3.2, Seite 21.

sind, d. h. sie müssen nicht nur die gleiche *Struktur* für den Datenverkehr verwenden, sondern auch ein gemeinsames *Verständnis* der Bedeutung der Daten haben.

Das Modell für den *information viewpoint* wird in ISO 19101 definiert. ISO 19101 definiert für den *information viewpoint* ein Modell namens *Extended Open Systems Environment (EOSE) für geographische Informationen*. Das Modell erweitert das *Open System Environment Model* (ISO/IEC TR 14252)[53] und definiert sechs Dienstkategorien:

Dienstkategorien

Interaktionsdienste für Nutzer (human interaction services) Dienste zur Verwaltung von Benutzungsschnittstellen, Graphiken, Multimedia und für die Präsentation zusammengesetzter Dokumente

Modell-/Informations-Verwaltungsdienste (model/information management services) Dienste zur Verwaltung der Entwicklung, Manipulation und Speicherung von Metadaten, konzeptuellen Schemata und Datensätzen

Arbeitsablauf-/Aufgabenverwaltungsdienste (workflow/task management services) Dienste zur Unterstützung von speziellen, von Personen kontrollierten Aufgaben

Verarbeitungsdienste (processing services) Dienste zur Ausführung von umfangreichen Berechnungen mit beträchtlichem Datenaufwand (weiter unterteilt in Dienste zur Verarbeitung räumlicher, thematischer, zeitlicher und Metadaten-betreffender Aspekte)

Kommunikationsdienste (communication servcies) Dienste zur Verschlüsselung und Übertragung von Daten über Kommunikationsnetzwerke

Systemverwaltungsdienste (system management services) Dienste zur Verwaltung von Systemkomponenten, Anwendungen und Netzwerken sowie Nutzerkonten und Zugriffsrechten

Taxonomie

Jede dieser Kategorien repräsentiert Dienste für einen bestimmten, semantisch unterscheidbaren Zweck. ISO/DIS 19119 führt *Beispiel-Dienste* für die Dienste-Kategorien an und definiert damit eine *Dienst-Taxonomie*. In dieser Klassifizierung findet sich unter den *Interaktionsdiensten für Nutzer* beispielsweise ein *catalogue viewer* zum Auffinden und Verwalten von Metadaten. Ein Beispiel für einen Modell-/Informations-Verwaltungsdienst ist der *feature access service*, der für den Datenzugriff und die Verwaltung gespeicherter Daten zuständig ist.

Namensgebung

Ein System muss diese Beispiel-Dienste nicht implementieren – sofern ein Dienst innerhalb eines Systems allerdings einen der aufgeführten Namen hat, sollte dieser Dienst auch die vordefinierte Funktionalität bereitstellen. Umgekehrt gilt Gleiches: ein Dienst mit der Funktionalität einer der Beispiel-Dienste

[53] ISO/IEC TR 14252: Information Technology – Guide to the POSIX Open System Environment (OSE).

sollte auch den in der Taxonomie vorgeschlagenen Namen haben. Die Namen aller Beispiel-Dienste der ISO/DIS 19119 sind im Anhang zu finden (Tabellen C.1/C.2, Seite 238/239). Die Definitionen der Funktionalität dieser Dienste sind gegebenenfalls in ISO/DIS 19119 (Percivall, 2002) nachzulesen.

Die Mehrzahl der Standards der 19100-Reihe fallen in die Klasse der *Modell-/Informations-Verwaltungsdienste* und der *Verarbeitungsdienste*. Für die Bereiche *Arbeitsablauf-/Aufgabendienste*, *Kommunikationsdienste* und *Systemverwaltungsdienste* gibt es in der 19100-Serie keine Standards. Das Kapitel der Architekturentwicklung (Kapitel 5, Seite 95) nimmt die Beispieldienste noch einmal auf.

Engineering Viewpoint

Um die Entwicklung eines flexiblen Systems zu unterstützen, werden die Dienste der oben genannten sechs Dienst-Kategorien in eine vierschichtige logische Architektur integriert (siehe Abbildung 3.10, Seite 62). Die erste (obere) Schicht ist zuständig für die physikalische Interaktion mit dem Nutzer und enthält daher alle *Interaktionsdienste* (human interaction services). Unter dieser Schicht liegen die *Verarbeitungsdienste*: in Schicht zwei die vom Nutzer geforderten Funktionen (user processing services) und in Schicht drei diejenigen Funktionen, die von mehreren Nutzern sowohl aus dem gleichen Anwendungskreis als auch bereichsübergreifend verwendbar sind (shared processing services). Die unterste Schicht beinhaltet die Dienste zur physikalischen Datenspeicherung und Datenverwaltung, also die *Modell-/Informations-Verwaltungsdienste* (model/information management services). Die *Arbeitsablauf-/Aufgabendienste* (workflow/task services) werden als spezialisierte Verarbeitungsdienste angesehen und stehen daher auf der Ebene der Prozessdienste. Die *Systemverwaltungsdienste* (system management services) können in jeder der vier Schichten auftreten (auch wenn sie in Abbildung 3.10 neben den Verarbeitungsdiensten platziert sind). Für die Verbindungen zwischen den Schichten sind die *Kommunikationsdienste* (communication services) zuständig.

4 logische Schichten

Zur Implementierung dieser logischen Architektur wird eine *physikalische Architektur* benötigt, d. h. eine Zuordnung der Dienste zu Komponenten und deren Schnittstellen. ISO/DIS 19119 schlägt eine zweischichtige und eine dreischichtige physikalische Architektur vor. Die zweischichtige besteht aus einer Komponente, die als Daten-Server dient, und aus einer Komponente, die den Clients und die Benutzungsschnittstelle implementiert. Bei der dreischichtigen physikalischen Architektur werden einige bzw. alle Verarbeitungsdienste aus der Client-Komponente entfernt und in einen Anwendungsserver verlagert. Sofern *alle* Verarbeitungsdienste vom Client in den Anwendungsserver überführt werden, wird der Client als *thin client* bezeichnet. Werden nur die von mehreren Nutzern verwendeten Funktionen in den Anwendungsserver verlagert und enthält der Client noch einen großen Funktionsanteil, handelt es sich um einen

2 oder 3 physikalische Schichten

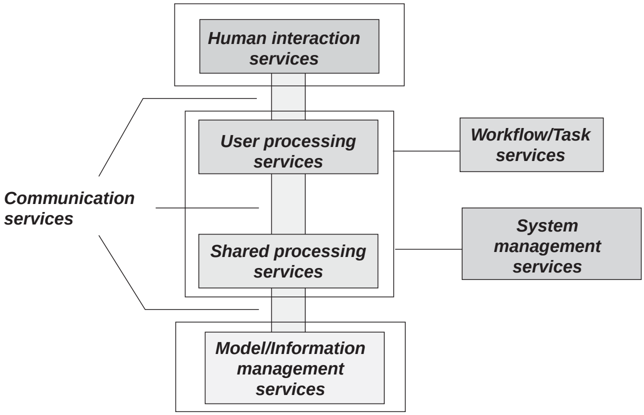


Abbildung 3.10: Logische Vier-Schichten-Architektur. Die Erklärung der Abbildung findet sich im Text. Quelle: ISO/DIS 19119 (Percivall, 2002).

thick client. Abbildung 3.11 zeigt die Zuordnung der logischen Schichten zu den physikalischen Schichten im Überblick.

Technology Viewpoint

Verteilungsplattform

Damit die Komponenten der physikalischen Architektur in einer verteilten Umgebung miteinander kommunizieren können, ist eine entsprechende Infrastruktur notwendig: eine so genannte *Distributed Computing Platform (DCP)*. DCPs erlauben die Zusammenarbeit der Komponenten über Rechnergrenzen, Hardware-Plattformen, Betriebssysteme und Programmiersprachen hinweg. Wichtige DCPs sind die *Common Request Broker Architecture (CORBA)* der Object Management Group (OMG), das *(Distributed) Common Object Model ((D)COM)* der Firma Microsoft und *Java Remote Message Invocation (JAVA RMI)* der Firma Sun Microsystems. Die Kommunikation zwischen Systemen, die mit unterschiedlichen DCPs realisiert wurden, kann mit spezieller Software (bridges) erfolgen.

Plattform-(un)abhängig

Da die verschiedenen DCPs unterschiedliche Vorgehensweisen zur Verteilungsunterstützung verwenden, müssen die genauen Dienste-Spezifikationen abhängig von der DCP entwickelt werden – auf diese Weise entstehen die so genannten *plattformspezifischen Dienste-Spezifikationen*. Die Basis für die Entwicklung einer plattformspezifischen Spezifikation bildet eine *plattformneutrale Spezifikation*, die unabhängig von der verwendeten DCP ist.

Beispiel

Ein Beispiel für dieses Vorgehen sind die Spezifikationen zu den geographischen Informationseinheiten (Features) des OpenGIS-Konsortiums: die platt-

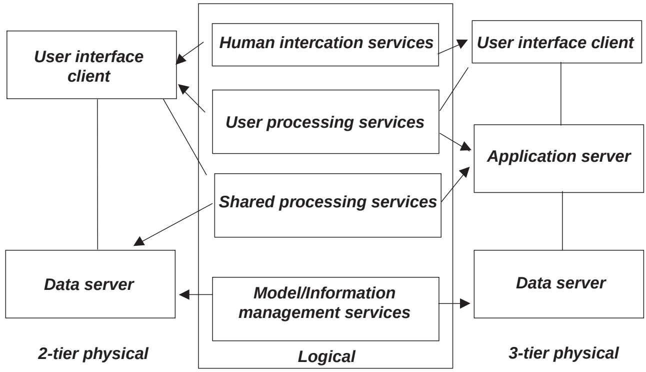


Abbildung 3.11: Physikalische Mehrschichten-Architektur. Die Erklärung der Abbildung findet sich im Text. Quelle: ISO/DIS 19119 (Percivall, 2002).

formneutrale Spezifikation findet sich in der *abstract specification* mit dem Titel *'The OpenGIS Feature'*, während sich die plattformspezifischen Definitionen für unterschiedliche DCPs in den *implementation specifications* befinden – unter ihnen beispielsweise eine Spezifikation für CORBA (OpenGIS Simple Features Specification for CORBA).[54]

3.3 Fazit

Integrierte Modelle

Für den Begriff des *'integrierten Modells'* gibt es keine einheitliche Definition. Mit dem Begriff wird in der Literatur sowohl die Verbindung mehrerer *konzeptioneller Modelle* zu einem Gesamtmodell verstanden als auch die Umsetzung der konzeptionellen Modelle in *Simulationsmodelle*, d.h. in ablauffähige *Software-Systeme*. Einige Definitionen gehen noch einen Schritt weiter, indem sie ein integriertes Modell als ein *umfangreiches Software-System* ansehen, das einen *konsistenten Rahmen* für die Simulationsmodelle bereitstellt und das neben der Simulation *weitere Funktionalitäten* anbietet. Eine Analyse existierender integrierter Modelle zeigt das Leistungsspektrum ihrer Software-Systeme, das sich nicht in der Berechnung neuer Simulationsergebnisse erschöpft: viele Systeme bieten Funktionen zur Visualisierung und statistischen Auswertung von Ergebnissen, Kopplungsmöglichkeiten zu Datenbanksystemen und GIS sowie weitere Leistungsmerkmale. Einige Systeme sind demnach weit mehr als

integriertes Modell

[54]Zum OpenGIS-Konsortium siehe Seite 45; eine Spezifikations-Übersicht findet sich in den Tabellen C.4 (Seite 241) und C.5 (Seite 242).

gekoppelte Teilmodelle und stellen in der Tat den angesprochenen Rahmen für die Simulationsmodelle bereit.

integriertes Modell vs. SISA

Um die unterschiedlichen Sichtweisen voneinander zu unterscheiden, wird in der vorliegenden Arbeit der Begriff des 'integrierten Modells' dann verwendet, wenn primär die miteinander gekoppelten Simulationsmodelle angesprochen werden. Wenn die Rahmen-Eigenschaft eines Systems betont werden soll, wird hingegen nicht von einem integrierten Modell gesprochen, sondern von einem 'System zum simulationsbasierten integrierten Assessment', das wie folgt charakterisiert wird: Ein *System zum integrierten simulationsbasierten Assessment (SISA)* ist ein Software-System, das von unterschiedlichen Fachdisziplinen stammende Daten und Simulationsmodelle zum 'System Erde' in einem konsistenten Rahmen kombiniert und neue Daten über den Zustand und mögliche langfristige Änderungen des 'Systems Erde' – vornehmlich zur Unterstützung politischer Entscheidungsträger – berechnet und bereitstellt.

Definition SISA

In Bezug auf die *Qualität* von Assessment-Ergebnissen spielt der 'konsistente Rahmen' eine entscheidende Rolle, da er zur Steigerung und Sicherstellung der Transparenz und Nachvollziehbarkeit der Ergebnisse beiträgt.

Architektur

Problem

Eine einheitliche Architektur der vorgestellten Systeme ist nicht zu finden. Die gestiegenen Leistungsanforderungen an SISAs (GIS-Integration, Kopplung verteilter Modelle etc.) spiegeln sich zwar in den einzelnen Systemen wider, eine *Komponentenbildung* im Sinne einer Software-Architektur (Einteilung des Gesamtsystems in die Hauptbestandteile und Definition von Verantwortlichkeiten und Schnittstellen) ist aber *nicht* zu finden. Die Systeme werden hingegen zumeist in *Module* eingeteilt, die sich aus der *Realisierung* der Systeme ergeben (z. B. in Klassen-Bibliotheken). Aufgrund der unterschiedlichen Funktionalitäten der Module, lassen sich diese nicht ohne weiteres unter den Systemen austauschen. Eine Interoperabilität zwischen den Systemen ist wegen der uneinheitlichen Einteilung der Gesamtsysteme sowie der unterschiedlichen Implementierungsmethoden bei der Funktionsrealisierung ebenfalls nicht gegeben.

Ansatz

Der Schlüssel zur Wiederverwendbarkeit und Interoperabilität sowie zu qualitativ hochwertigen Systemen befindet sich in der Definition einer Software-Architektur. Informationen zu den *Leistungsmerkmalen*, *System-Aufteilungen* und *Implementierungen* existierender Modelle und Werkzeuge können als Grundlage für die Architektur-Entwicklung für SISAs herangezogen werden. Zur Steigerung der Interoperabilität von Systemen zum simulationsbasierten integrierten Assessment sollten bei deren Entwicklung auch *Architektur-Standards* berücksichtigt werden. Aufgrund der Komplexität der Standards werden standardkonforme Realisierungen von SISAs i. d. R. nicht zu erwarten sein. Die Architektur eines SISA sollte sich dennoch an diesen Standards orientieren, da bereits die Umsetzung ihrer *Prinzipien* eine verbesserte Qualität der

Systeme verspricht. Darüber hinaus erlaubt die Berücksichtigung der grundlegenden Prinzipien der Standards eine *schrittweise Migration* zu standardkonformen Systemen.

Die Entwicklung einer an Standards orientierten Architektur ist Inhalt des Kapitels 5 (Seite 95). Grundlage für die Entwicklung der Architektur sind u. a. die allgemeinen Ziele und Anforderungen eines SISA, die im folgenden Kapitel definiert werden.

Wesentliche Aspekte zu *Standards*, den *Leistungsmerkmalen* sowie der *Aufteilung* und *Implementierung* existierender Systeme werden zum Abschluss dieses Kapitels in den folgenden Absätzen noch einmal zusammengefasst.

Leistungsmerkmale

Die folgenden zentralen Leistungsmerkmale bzw. Funktionalitäten gehören, neben der Berechnung neuer Simulationsergebnisse, zum Leistungsumfang der integrierten Modelle und sind daher bei der Erstellung der Software-Architektur für ein SISA zu berücksichtigen:

- Bereitstellung und Nutzung von GIS-Funktionalitäten
- Unterstützung bei der Analyse von (Ergebnis-)Daten
- Visualisierung von Modellergebnissen
- Unterstützung bei der Modellanalyse
- Verwaltung von Szenarien
- Bereitstellung und Erstellung von Dokumentationen (Modelldokumentation, Metadaten, Hilfesystem)
- Verwaltung von Datenbeständen

GIS

Die GIS-Funktionalitäten werden für die geographisch explizite Modellierung des 'Systems Erde' benötigt und werden entweder direkt im Rahmen des integrierten Modells, also im SISA, implementiert oder durch die Kopplung mit einem eigenständigen GIS in das System integriert. Die Verwendung bzw. Integration von GIS-Funktionen finden sich sowohl bei vielen der von Kickert u. a. (1999) und Peirce (1998) beschriebenen Modelle wieder als auch bei den Systemen *OMS*, *IMA* und *SNI*.

Datenanalyse

Zur Unterstützung der Analyse von Daten und Ergebnissen werden von den Systemen beispielsweise Funktionen zur statistischen Auswertung bereitgestellt. Die bei Peirce (1998) aufgeführten Systeme stellen Funktionen zur Berechnung von Standard-Statistiken (Mittelwert, Standardabweichung, kumulative Wahrscheinlichkeit etc.) und zur Analyse von Zeitreihen bereit. Die Möglichkeit zur Visualisierung von Ergebnissen in unterschiedlicher Form (Tabellen, Graphen, Karten) kann als einfache Art der Analyseunterstützung angesehen werden und wird sowohl bei den Systemen *DANUBIA*, *IMA*, *M* und *OMS* als auch bei vielen weiteren Systemen (vgl. Kickert u. a., 1999; Peirce, 1998) erwähnt.

Modell-analyse

Neben der Analyse von Modellergebnissen werden von einigen Systemen auch Funktionen zur Analyse und zum Betrieb des Modells bereitgestellt. Hierzu gehören insbesondere Funktionen zur Unsicherheitsanalyse aber auch Möglichkeiten zur Modellkalibrierung und Modelloptimierung (vgl. die Ziele von PRISM sowie Kickert u. a., 1999; Peirce, 1998).

Szenarien-Verwaltung

Die Verwaltung von Szenarien bzw. Simulationsläufen wird – trotz der Wichtigkeit für die Nachvollziehbarkeit und Transparenz von Simulationsergebnissen – bei den meisten Systembeschreibungen nicht erwähnt. Für das Modellierungssystem *M* wird hingegen eine separate Komponente hierfür vorgeschlagen (der 'scenario manager'), über die verschiedene Szenario-Einstellungen gesichert und miteinander verglichen werden können. Für das System *DANUBIA* steht ebenfalls eine Komponente zur Verwaltung komplexer Szenarien auf dem Entwicklungsplan.

Dokumentation

Die Dokumentation von Modellen bzw. Ergebnissen, die für die Wiederverwendbarkeit äußerst relevant ist, spielt eine sehr unterschiedliche Rolle bei den betrachteten Systemen: während die meisten Systembeschreibungen nicht auf dieses Thema eingehen, beschäftigen sich bei *GLOBESIGH* gleich zwei der vier Komponenten mit der Dokumentation von Hintergrund- und Ergebnisinformationen (die 'information base' und die 'issues base'). Bei *ECOBAS* fußt die (automatische) Modellgenerierung sogar auf einer detaillierten Beschreibung der Modellelemente. Hier ist die Dokumentation also direkt mit den Modellen gekoppelt. Darüber hinaus besitzt *ECOBAS* zur Steigerung der Wiederverwendbarkeit von Modellen ein sehr detailliertes Metadaten-Konzept. Die Integration von Metadaten spielt bei den Systemen ansonsten – wenn überhaupt – eine sehr untergeordnete Rolle. Peirce (1998) führt in seiner Übersicht auch Systeme an, die die Generierung von Ergebnis-Berichten unterstützen. Das System *M* bietet, neben einem Hilfesystem, die Möglichkeit der Hypertext-basierten Berichterstattung.

Datenverwaltung

Bei den Beschreibungen integrierter Modelle spielt die Verwaltung von Datenbeständen zumeist eine untergeordnete Rolle. Die Integration von Daten in die Systeme geschieht mit unterschiedlichen Mitteln: während *GLOBESIGH* ein Datenbank-Managementsystem verwendet, stellt das System *IMA* eigene Funktionen zum Datenimport und -export bereit und kapselt Daten in so genannten 'Datenmodulen'. *OMS* kapselt die Daten in Bibliotheken und separiert die Parameter in einer eigenen Bibliothek. Die Notwendigkeit der Integration von Datenbanken hängt laut Peirce (1998) von der Höhe des Entwicklungsstandes eines Modells ab. Während einige Modelle Datenbanken lediglich als Datenlieferanten nutzen, stellen bei anderen Modellen ausgereifte Datenbank-Managementsysteme den Kern des Systems dar.

Komponenten

Bei den Systemen, die mehr als die Simulationsmodelle beinhalten, lassen sich die folgenden Systemkomponenten abgrenzen: Komponenten

- Simulationsmodell
- Dokumentation
- Datenhaltung
- Datenverwaltung
- Datenintegration
- Ablaufsteuerung

Die Trennung des Simulationsmodells von den übrigen Teilen des Systems wird sowohl bei *GLOBESIGHT* und *M* als auch bei *OMS* und *IMA* vorgenommen. Die Funktionen zur Dokumentation können insbesondere bei den Systemen *GLOBESIGHT*, *M* und *ECOBAS* als separate Komponenten betrachtet werden. Eine klare Trennung von Datenhaltung und Datenverwaltung (inkl. einem Szenario-Manager) findet sich insbesondere beim Modellentwicklungssystem *M*. Separierte Funktionen zum Datenexport und -import (Datenintegration) sind bei den Systemen *OMS*, *GLOBESIGHT* und *IMA* vorhanden. *OMS* und *M* bieten zur Verarbeitung von Daten außerdem eine Art 'Ablaufsteuerung', mit der die Funktionen des Systems programmgesteuert angesprochen werden können.

Die größte Aufmerksamkeit bei der Beschreibung integrierter Modelle erhält die Kopplung von Modellteilen. Hier existieren die unterschiedlichsten Ansätze: angefangen von relativ einfach zu realisierenden Konzepten (wie bei *SNI*) bis zu sehr spezialisierten Modellkopplern (wie bei *PRISM*). Modellkopplung

Implementierung

Bei den eingesetzten Mitteln zur Implementierung integrierter Modelle zeigt sich eine Tendenz zur Verwendung *objektorientierter Technologien* sowie zur Nutzung von *Internet-Technologien*. So ist das *OMS* beispielsweise in *JAVA* implementiert und nutzt *Remote Message Invocation (RMI)* zur Verteilung sowie die *JAVA-NetBeans* zur Komponentenkapselung. *DANUBIA* ist ebenfalls in JAVA implementiert. Da über das *Java Native Interface (JNI)* die Integration von Programmen möglich ist, die in anderen Programmiersprachen geschrieben sind, wird die Integration von in C geschriebenen Programmen erlaubt (auch FORTRAN-Programme lassen sich über den Umweg über C-Programme einbinden).

Die Nutzung von Internet-Browsern als Benutzungsschnittstelle findet sich sowohl in aktuellen Entwicklungen wie *DANUBIA* als auch von den 'älteren' Systemen *M* und *ECOBAS*. Das System *IMA* verwendet darüber hinaus die *extensible markup language (XML)*.

Standards

Im Zusammenhang mit der Entwicklung einer Software-Architektur sind insbesondere die Arbeiten des *technischen Komitees 211 (TC211)* der *internationalen Organisation für Standardisierung (ISO)* sowie die Arbeiten des *Open GIS Consortium (OGC)* von Bedeutung. Sowohl das TC211 als auch das OGC liefern Standards zum Aufbau von Systemen zur Geodatenverarbeitung. Den gemeinsamen *Rahmen* für diese Standards bildet der *ISO/DIS 19119*. Dieser Standard beschreibt den grundsätzlichen architektonischen Aufbau, den geodatenverarbeitende Systeme aufweisen sollten. Für die *Kopplung* komplex miteinander verknüpfter Teilmodelle sollte die *High-Level Architecture (HLA)* des *Institute of Electrical and Electronic Engineers (IEEE)* in Betracht gezogen werden. Die Verwendung von Standards des *World Wide Web Consortium (W3C)* bei der Realisierung von SISAs verspricht ebenfalls eine gesteigerte Interoperabilität der Systeme.

Kapitel 4

Systemdefinition

Ziel der vorliegenden Arbeit ist die Entwicklung einer Software-Architektur, die als Referenz bei der Erstellung und Erweiterung von Systemen zum integrierten simulationsbasierten Assessment (SISAs) verwendet werden kann. Eine Voruntersuchung und Durchführbarkeitsstudie, wie sie in der Planungsphase einer Software-Entwicklung vorgesehen ist[1], kann an dieser Stelle aufgrund der gewünschten Anwendbarkeit der Architektur auf unterschiedliche Systeme nicht vorgenommen werden (die genauen Funktionen eines SISA sind projektabhängig). Dieses Kapitel beschäftigt sich daher mit der Definition *allgemeiner Anforderungen* an ein System zum integrierten simulationsbasierten Assessment und dient damit als Grundlage für die Architekturentwicklung in Kapitel 5 (Seite 95).

Grundlage für das Verständnis der Anforderungsdefinition ist das in Abschnitt 4.1 spezifizierte Produktmodell, das wichtige Aspekte des Anwendungsbereichs erklärt. Abschnitt 4.2 (Seite 75) definiert anschließend das Einsatzgebiet und die Hauptfunktionen und Hauptdaten des SISA sowie die grundlegenden Anforderungen an das System. Die wichtigsten Eigenschaften und Anforderungen des spezifizierten SISA werden in Abschnitt 4.3 noch einmal zusammengefasst. Übersicht

4.1 OOA-Modell

Inhalt dieses Abschnitts ist die Darstellung und Erklärung des Produktmodells, das aus einer objektorientierten Analyse des Problembereichs resultiert (OOA-Modell). Der erste Unterabschnitt (4.1.1) beschäftigt sich mit den wichtigsten Elementen eines SISA und mit der Umgebung, in die das System im

[1] Vergleiche Unterabschnitt 2.3.1, Seite 16.

Einsatz eingebettet ist (Gesamtmodell). Die Identifizierung verschiedener, wieder verwendbarer 'Betriebsmittel' (Ressourcen), die im Rahmen eines simulationsbasierten Assessments benötigt werden, erfolgt aufgrund ihrer Wichtigkeit ausführlicher im anschließenden Unterabschnitt (4.1.2, Seite 72).

4.1.1 Gesamtmodell

Die in Abbildung 4.1 (Seite 73) dargestellten Objekte geben einen ersten Überblick über wichtige Bestandteile und Konzepte, die im Rahmen eines SISA von Bedeutung sind. Die folgenden Absätze dienen der Erklärung der Abbildung.

Simulations-modell

Das zentrale Element in einem SISA (s. Abb. 4.1, Seite 73) ist das integrierte Simulationsmodell, das sich i. d. R. aus mehreren Teilmodellen zusammensetzt. Die Teilmodelle repräsentieren dabei die verschiedenen Sphären, die zur Problembetrachtung notwendig sind. Oft werden bereits vorhandene Modelle oder vereinfachte Versionen bereits entwickelter Modelle (so genannte Metamodelle) innerhalb eines integrierten Modells als Teilmodell eingesetzt. Die Modellteile sind dabei nicht zwangsläufig von einem Entwickler oder einem Entwicklungsteam innerhalb einer Organisation erstellt worden – bedingt durch die hohe Komplexität der Modellteile werden diese oft weltweit von unterschiedlichen Organisationen wieder verwendet.

Implementierung

Die Implementierung der Teilmodelle eines integrierten Modells kann generell mit Hilfe verschiedener Simulationsansätze erfolgen. Im Bereich der integrierten Modellierung wird i. d. R. die Eigenentwicklung unter Verwendung von Standard-Programmiersprachen, wie FORTRAN, C, C++, oder speziell entwickelten Simulationsumgebungen bevorzugt. Ein Grund für dieses Vorgehen sind die unzureichenden Möglichkeiten kommerzieller Simulationsumgebungen und Simulationssprachen zur geographisch expliziten Repräsentation von Vorgängen. Die Kopplung der Teilmodelle erfolgt über einen Kommunikationsmechanismus; im einfachsten Falle mit Hilfe von Dateien (lose Kopplung) oder durch direkte Funktionsaufrufe (enge Kopplung) der beteiligten Teilmodelle.

Daten

Integrierte Modelle benötigen zur Kalibrierung und zum Betrieb große Datenmengen; vornehmlich Zeitreihen von Daten mit räumlichem Bezug (Geodaten). Die Datensätze liegen meist in Form von Dateien vor, die zur Simulationszeit direkt vom Simulationsmodell gelesen werden. Bei den verwendeten Datenformaten kommen sowohl Standardformate als auch proprietäre Formate vor. Die Speicherung der Daten kann auch über Datenbank-Managementsysteme erfolgen. Bei den Geodaten kann unterschieden werden zwischen solchen mit primärem Raumbezug und solchen mit sekundärem Raumbezug[2].

[2]Zum primären Raumbezug werden primäre Metriken verwendet, bei denen zwei- oder dreidimensionale Koordinaten anzugeben sind. Die Angabe geographischer Koordinaten (x- und y-Wert) ist ein Beispiel für einen primären Raumbezug. Der primäre Raumbezug erlaubt eine sehr genaue und eindeutige Ortsangabe. Sekundäre Metriken, wie beispielsweise die Postleitzahl-Bereiche oder Länderkennungen, werden für Ortsangaben verwendet, die eine

Datensätze können ferner durch ihre Herkunft differenziert werden: Die Datensätze, die von einem Datenlieferanten kommen, für deren Inhalt also nicht der Modellbetreiber oder Modellentwickler verantwortlich ist, können als ‘Primärdatensätze’ bezeichnet werden. Daten, die von Primärdatensätzen abgeleitet (z. B. modifiziert oder umformatiert) sind, können als ‘Sekundärdatensätze’ bezeichnet werden.

Simulationen (Simulationsläufe) basieren auf in sich konsistenten Annahmen über zukünftige Entwicklungen (den Szenarien), die über Datensätze und weitere Modelleinstellungen (z. B. bezüglich der Modellkonfiguration) quantifiziert werden[3]. Szenarien

Die Vor- und Nachbearbeitung von Daten geschieht sowohl über eigens für diesen Zweck geschriebene Programme (Individualsoftware) als auch über Standardsoftware (Textverarbeitung, Tabellenkalkulation etc.). Das Simulationsprogramm nutzt i. d. R. lediglich die Individual-Software zur Erledigung seiner Aufgabe; ein Trend zur Integration der Funktionalität von Geo-Informationssystemen (GIS) ist allerdings zu erkennen. Werkzeuge

An einem integrierten Assessment ist nicht nur eine Person beteiligt. Es kann unterschieden werden zwischen: Modellentwickler, Modell-Implementierer, Modell-Betreiber, Entscheidungsträger und Ressourcenlieferant. Der Modellentwickler ist zuständig für die Formulierung des konzeptionellen Modells und die Analyse des Simulationsmodells; die Umsetzung des konzeptionellen Modells in ausführbaren Computercode, d. h. die Erstellung des Simulationsmodells, ist Angelegenheit des Modell-Implementierers; für die Bereitstellung von Daten und anderen Ressourcen (z. B. Teilmodelle oder Hilfsprogramme) sind verschiedene Ressourcen-Lieferanten zuständig und der Modellbetreiber führt die Simulationsstudie durch und vermittelt die Ergebnisse den Entscheidungsträgern. Die beschriebenen Aufgaben können auch von weniger Beteiligten in Personalunion erledigt werden. Aufgrund der Komplexität integrierter Modelle sind an deren Entwicklung oft Personen aus unterschiedlichen Institutionen (Organisationen) beteiligt. Personen

Die an einem Assessment beteiligten Personen erzeugen verschiedene Dokumente. Bei den Dokumenten kann unterschieden werden zwischen formlosen Beschreibungen, kurzen Anmerkungen und umfangreicheren Berichten. Bei den für die Publikation von Ergebnissen erzeugten Dokumenten kann unterschieden werden zwischen Präsentationen (z. B. in Form von ‘Vortragsfolien’), Postern, Karten und veröffentlichter Literatur. Dokumente

Das Assessment selbst beinhaltet eine Simulationsstudie, zu der die Durchführung mehrerer Simulationsläufe und deren Analyse gehört. Die Simulations- Assessment

---

größere Unschärferelation aufweisen dürfen. Näheres zu diesem Thema findet sich z. B. in Bill und Fritsch (1994).

[3]Der Begriff des ‘Szenarios’ wird aus Gründen der Einfachheit auch bei der Zusammenfassung von Daten der Vergangenheit (die beispielsweise zur Modellvalidierung benötigt werden) benutzt.

läufe basieren auf definierten Szenarien. Eine Modellanalyse (beispielsweise zur Untersuchung der Modellunsicherheiten oder -sensitivität) gehört ebenfalls zu einem simulationsbasierten integrierten Assessment.

4.1.2 SISA-Ressourcen

Ressourcen

Die Erstellung einer Studie mit Hilfe eines SISA erfordert die Verwendung vieler unterschiedlicher ‘Betriebsmittel’ (Ressourcen[4]). Die Erzeugung dieser Betriebsmittel ist aufgrund der Komplexität eines integrierten Assessments sehr aufwendig – eine Wiederverwendung der Ressourcen ist also anzustreben. Die Frage ist daher: Welche Ressourcen können innerhalb des SISA identifiziert werden und wie kann die Wiederverwendbarkeit dieser Ressourcen erreicht oder gesteigert werden?

Identifizierung

Die Identifizierung der Ressourcen wird anhand des OOA-Modells in Abb. 4.1 unter Berücksichtigung der Arbeitsschritte erfolgen, die zur Erstellung einer Simulationsstudie notwendig sind (vgl. Abschnitt 2.2, Seite 14). Tabelle 4.1 zeigt die zur Erstellung einer Simulationsstudie erforderlichen Arbeitsschritte und deren Resultate in einer Übersicht.

Arbeitsschritt	Resultat
Problemformulierung und -analyse	Systemgrenzen, Zielbeschreibung
Modellbildung	Modellkonzept
Datenerhebung	Datenbasis
Erstellung des Computerprogramms	Simulationsmodell
Modellvalidierung	validiertes Simulationsmodell
Planung und Durchführung von Simulationsläufen	Simulationsergebnisse
Auswertung und Implementierung der Ergebnisse	Analysen, Berichte

Tabelle 4.1: Arbeitsschritte einer Simulationsstudie (vgl. Abschnitt 2.2, Seite 14) und deren Resultate. Die Resultate können als wieder verwendbare Ressourcen angesehen werden. Weitere Ressourcen sowie deren Unterteilung werden im Text aufgeführt.

Modellentwicklung

Zur Durchführung einer *Simulationsstudie (Studie)* ist demnach zunächst ein *Modellkonzept* zu erstellen. Dieses Modellkonzept ist bereits eine Ressource, die (bei entsprechend guter Dokumentation) in anderen Projekten sinnvoll wieder verwendet werden kann. Auf der Grundlage des Modellkonzepts wird das *Simulationsmodell* erstellt. Dieses *Gesamt-Simulationsmodell* besteht im Rahmen des integrierten Assessments i. d. R. aus unterschiedlichen, dem Modellkonzept folgenden *Teilmodellen*. Sowohl das Gesamt-Simulationsmodell als

[4]„Bestand von etwas, was für einen bestimmten Zweck benötigt wird.“ (Duden, 1996)

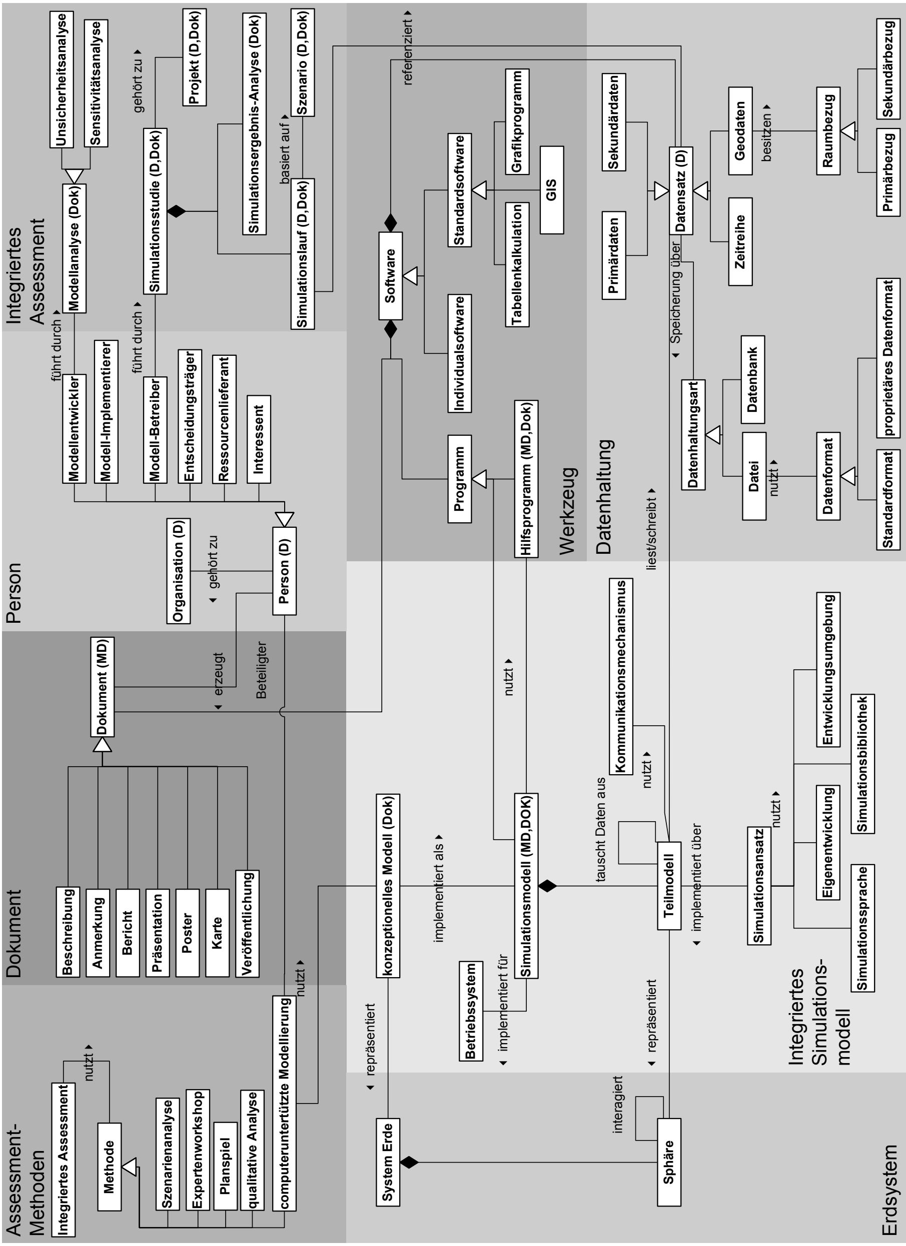


Abbildung 4.1: OOA-Modell eines Systems zum simulationsbasierten integrierten Assessments (SISA). Zur besseren Übersicht sind nicht alle Beziehungen zwischen den Klassen aufgeführt (vgl. auch Abb. 4.4, Seite 85).

auch die Teilmodelle stellen Ressourcen dar, deren Wiederverwendung aufgrund des Entwicklungsaufwandes in besonderem Maße anzustreben ist.

Daten u. Szenarien

Die Simulation des Systemverhaltens erfordert die Bereitstellung von Daten für das Simulationsmodell: *Initialisierungsdaten* für das Simulationsmodell, *Eingabedaten* zur Beschreibung exogener, systembeeinflussender Größen (Modellumwelt-Daten), *Parametersätze* zur Festlegung der Systemparameter sowie Angaben über Simulationsmodell-*Optionen*[5]. All diese Daten sind, ebenso wie die *Ausgabedaten* des Simulationsmodells, als wichtige SISA-Ressourcen zu betrachten. Die Simulationsläufe basieren auf Szenarien; um Ergebnisse verschiedener Assessments vergleichen zu können, werden für unterschiedlicher Studien und Projekte oft die gleichen Szenarien benutzt[6] – ein Grund, auch diese als SISA-Ressourcen zu betrachten.

Hilfsprogramme

Zur Erzeugung und Vorbereitung der Eingabedaten für die Simulationsmodelle werden sowohl Standardprogramme (*Anwendungen* wie GIS oder Tabellenkalkulationsprogramme) als auch eigens für diesen Zweck geschriebene Programme (Individualsoftware) eingesetzt. Die mehr oder weniger komplexen eigens entwickelten Programme können als *Hilfsprogramme* betrachtet und als Ressource innerhalb anderer Projekte wieder verwendet werden. Gleiches gilt für Funktionen, die z. B. in Form von 'Diensten' (s. Seite 56) angeboten werden.

Dokumente

Die Anwendungen und Werkzeuge werden auch bei der *Analyse* der Simulationsergebnisse eingesetzt – zunächst für die Validierung des Modells im *Modelltest* und anschließend für die *Szenarien-Analyse*. Endergebnis einer Studie sind dann verschiedene *Dokumente* wie Berichte und Veröffentlichungen (vgl. Abb. 4.1, Seite 73). Diese Dokumente sollten, ebenso wie die Beschreibung der Modellziele und der Modellgrenzen aus dem ersten Arbeitschritt, möglichst einfach zugreifbar und wieder verwendbar sein.

Während der Erstellung des Modells und der Durchführung der Studie treten zahlreiche Fragen und Probleme auf, deren Lösungen eventuell auch für weitere Arbeiten hilfreich sind; kurze, formlose Anmerkungen können hier eine entscheidende Erleichterung bei der Wiederbenutzung der anderen Ressourcen bieten. Aus diesem Grund wird die *Anmerkung* explizit als Ressource angesehen.

abstrakte Ressourcen

Zur Beschreibung von Simulationsläufen, Studien, Projekten etc. sind Dokumente zu erstellen. Diese Dokumente können, ebenso wie Daten oder Software, unmittelbar für andere Zwecke wieder verwendet werden. Ein Simulationslauf an sich kann aber auch eine Ressource darstellen; eine Ressourcen, von denen Teile (z. B. Ausgabedaten) wieder verwendet werden können oder auf die an anderen Stellen verwiesen werden kann. Um solche Verweise zu erlauben, werden für das SISA so genannte *'abstrakte Ressourcen'* eingeführt.

[5]Beispielsweise zur Steuerung des Umfangs oder Formats der Ausgabedaten.

[6]Beispielsweise kann im Rahmen der Klimafolgenforschung die Berücksichtigung von Szenarien des *IPCC Special Report on Emission Scenarios* (IPCC, 2000) als obligatorisch angesehen werden.

Die (abstrakten) Ressourcen, die sich teilweise auch im objektorientierten Analysemodell des SISA auf Seite 73 wiederfinden, sind in Abbildung 4.2 (Seite 76) als Übersicht dargestellt.

4.2 Anforderungsdefinition

Im Folgenden werden die funktionalen und nicht-funktionalen Anforderungen an ein SISA beschrieben. Die Form der Anforderungsdefinition orientiert sich am Vorschlag zum Aufbau eines Pflichtenheftes aus Balzert (1996). Da die funktionalen Anforderungen an ein SISA von System zu System variieren, werden an dieser Stelle lediglich die Grundfunktionalitäten aufgeführt, die für jedes SISA angenommen werden können.

Der folgende Unterabschnitt (4.2.1) beschreibt zunächst *allgemeine Anforderungen* an das SISA, wie sie in der einschlägigen Literatur zu finden sind. Daraufhin erfolgt die Definition der *Ziele und Funktionen* des SISA (Unterabschnitt 4.2.2, Seite 78). Der Anwendungsbereich, die Zielgruppen und die Betriebsbedingungen werden im Unterabschnitt *System-Einsatz* (4.2.3, Seite 80) beschrieben. Anschließend erfolgt die Beschreibung der *System-Umgebung* (4.2.4, Seite 82) und der grundlegenden *System-Daten* (4.2.5, Seite 84). Besondere Anforderungen an zeit- und umfangbezogene Leistungen des SISA werden im Unterabschnitt der *System-Leistungen* (4.2.6, Seite 88) behandelt. Anforderungen an die *Benutzungsschnittstelle* finden sich im folgenden Unterabschnitt (4.2.7, Seite 89). Ein SISA hat nicht nur funktionale Anforderungen zu erfüllen, sondern auch nicht-funktionale (qualitative); diese werden im Abschnitt *Qualtitätsziel-Bestimmung* (4.2.8, Seite 89) aufgeführt. Im Anschluss folgen Unterabschnitte zur Beschreibung der *Testszenarien* und der *Entwicklungs-Umgebung* sowie ein kurzer Abschnitt zu *Ergänzungen*.

Übersicht

4.2.1 Allgemeine Anforderungen

Das in einem SISA verwendete Simulationsmodell ist i. d. R. kein monolithisches Programm, sondern ist aus unterschiedlichen (oft über Institutionsgrenzen hinweg erstellten) Einzelmodellen zusammengesetzt. Selbst Modelle, die bereits im Einsatz sind, werden alle drei bis sechs Monate verfeinert und auf einen neuen Stand gebracht (Weyant u. a., 1996), d. h. ein SISA bzw. das Simulationsmodell innerhalb des SISA muss gut änderbar sein.

Modelländerungen

McCarthy u. a. (2001) fordern die Nutzung prozessorientierter Modelle mit hoher räumlicher und zeitlicher Auflösung, was die Nutzung entsprechend aufgelöster (geographischer) Daten nach sich zieht. Zur Vorbereitung dieser Daten für die Simulationsmodelle, die anschließende Analyse der Simulationsergebnisse und die Analyse von Hintergrunddaten ist der Einsatz von Geo-Infor-

Geo-Daten

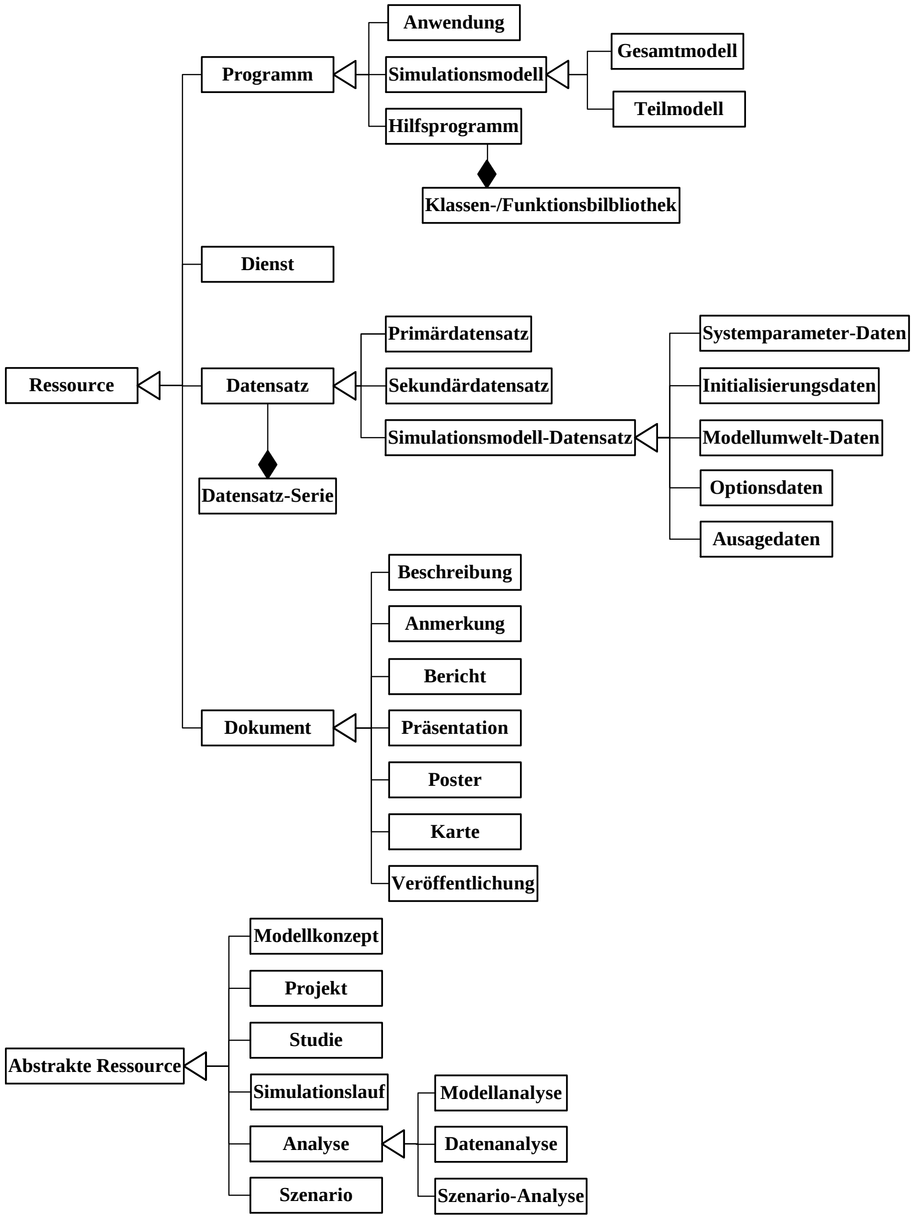


Abbildung 4.2: Ressourcen eines Systems zum simulationsbasierten integrierten Assessment. Erklärungen finden sich im Text. (s. auch Abb. 4.1, Seite 73 u. Abb. 4.4, Seite 85).

mationssystemen sinnvoll bzw. notwendig (s. z. B. Luiten, 1999; Peirce, 1998; Schneider, 1997).

Visualisierung

Da nicht alle Daten sinnvoll in einem GIS angezeigt werden können, müssen entsprechende Visualisierungsmöglichkeiten vorhanden sein (Alcamo, 2002; Peirce, 1998; Rotmans und Dowlatabadi, 1998)[7]. Zur Darstellung sollten nicht nur Diagramme, Graphen, Tabellen und Histogramme genutzt werden, sondern auch innovative Techniken, z. B. zur dynamischen Präsentation von Information (Rotmans und Dowlatabadi, 1998).

Datenverwaltung

Zur Erstellung simulationsbasierter Assessments werden umfangreiche Datenbestände benötigt und erzeugt. Die Organisation und Verwaltung dieser Daten (inklusive dazugehöriger Datendokumentationen) spielt eine wichtige Rolle im integrierten Assessment (Luiten, 1999; Parson, 1995) und sollte daher vom SISA unterstützt werden. Je nach Komplexität der konkreten Anwendung ist auch die Anbindung an ein Datenbank-Managementsystem sinnvoll bzw. notwendig (Luiten, 1999).

Nachvollziehbarkeit

Alcamo (2002) fordert im Hinblick auf die Anwendbarkeit von integrierten Modellen eine gute Modelldokumentation. Eine umfangreiche Dokumentation ist auch notwendig zur von Toth (1995) geforderten Modell-Verifikation durch den Vergleich von Ergebnissen unterschiedlicher Modelle: um einen solchen Vergleich auch über Institutionsgrenzen hinweg zu ermöglichen, müssen sowohl die einem Simulationslauf zugrunde liegenden Szenarien als auch die verwendeten Basisdaten vergleichbar und gut dokumentiert sein. Die Dokumentation aller für einen Simulationslauf verwendeten Daten erhöht darüber hinaus die von Dowlatabadi (1995) und Rotmans und Dowlatabadi (1998) geforderte Transparenz des Modells.

Austauschbarkeit

Als weiteren Weg zur Modell-Überprüfung schlägt Toth (1995) den Austausch von Modellteilen mit anschließendem Ergebnisvergleich vor. Neben der Notwendigkeit einer ausreichenden Dokumentation bedarf es für diesen Schritt einer guten 'Austauschbarkeit' von Teilmodellen.

Modell-Analyse

Die Modell-Validierung sollte sich nach Rotmans und Dowlatabadi (1998) nicht nur auf den Ergebnis-Vergleich unterschiedlicher integrierter Modelle für einen definierten Satz von Szenarien beschränken. Die Autoren fordern auch einen Vergleich endogener und exogener Modellannahmen und die Durchführung umfassender Sensitivitäts- und Unsicherheitsanalysen. Auch, wenn Weyant u. a. (1996) sagen, dass eine Sensitivitäts-Analyse über ein gesamtes integriertes Modell aus Komplexitätsgründen nicht möglich ist und Alcamo (2002) anmerkt, dass sich die Unsicherheiten einzelner Teilmodelle unter gewissen Umständen auch kompensieren können, werden umfassende Untersuchungen zur Sensitivität und Unsicherheit an vielen Stellen gefordert (Alcamo, 2002; Cocks u. a., 1998; Dowlatabadi, 1995; Rotmans und van Asselt, 2001;

[7] „Readers will prefer it if tools that generate data provide their own visualisation functions suited to the data.“ (Peirce, 1998)

Schneider, 1997; Shlyakhter u. a., 1995; Toth, 1995; Tol und Vellinga, 1998). Eine Unterstützung der Modellanalyse durch das SISA ist daher vorzusehen.

Unter Berücksichtigung dieser allgemeinen Anforderungen werden im folgenden Unterabschnitt die vom SISA verfolgten Ziele genauer definiert.

4.2.2 Ziele und Funktionen

Das übergeordnete Ziel des Systems zum integrierten simulationsbasierten Assessment ist die konsistente Verwaltung von Daten unterschiedlicher Fachdisziplinen und die Erzeugung neuer Daten mit Hilfe von Simulationsmodellen.

/Zxx/ /Fxx/

Im Folgenden werden die Ziele des Systems sowie die Funktionen, die notwendig sind, um das SISA-Ziel zu erreichen, genauer definiert. Um im weiteren Verlauf der Arbeit einfacher auf diese Definitionen zurückgreifen zu können, werden die einzelnen Ziele und Funktionen – dem Vorschlag von Balzert (1996) folgend – bezeichnet: Die Ziele werden mit /Zxx/ bezeichnet, Funktionen mit /Fxx/. Die Nummerierung (xx) erfolgt dabei in 10er-Schritten, um nachträgliche Änderungen zu vereinfachen.

Simulationslaufverwaltung

Um die geforderte Nachvollziehbarkeit von Modellergebnissen und Studien zu gewährleisten, muss das SISA in der Lage sein, Simulationsläufe (Simulationen) zu verwalten (/Z10/). Zur Simulationsverwaltung muss das System eine Funktion zur Beschreibung neuer Szenarien bereitstellen (/F10/). Einem Szenario werden i. d. R. mehrere Datensätze zugeordnet (z. B. über den Verlauf wichtiger Indikatoren; vgl. Abb. 4.1, Seite 73). Das System muss daher in der Lage sein, eine solche Zuordnung aufzunehmen (/F20/). Weitere, zur Berechnung von Modellergebnissen notwendige Daten (z. B. zur Parametrisierung des Modells) müssen den einzelnen Simulationsläufen zugeordnet werden können. Entsprechende Funktionen zur Beschreibung von Simulationsläufen (/F35/) und der Datenzuordnung (/F30/) müssen daher vom SISA bereitgestellt werden. Um die Auswahl von Daten zu unterstützten, sollte das SISA Informationen (Metadaten) über vorhandene bzw. anwendbare Datensätze bereitstellen (/F40/).

Projekt-Ressourcen

Über die Verwaltung der Simulationsläufe hinaus soll das SISA in der Lage sein, alle Ergebnisse einer Simulationsstudie (inkl. der auf den Modellrechnungen basierenden Analysen) zu verwalten (/Z20/). Zu diesem Zweck ist eine Funktion bereitzustellen, die eine Zuordnung von Ressourcen zu Simulationsstudien und Projekten erlaubt (/F50/).

Ressourcenliste

Die Verwaltung der Ressourcen selbst wird ebenfalls als ein Ziel des SISA definiert (/Z30/). Zur Steigerung der Wiederverwendbarkeit von Ressourcen sollte zumindest eine Funktion zur Auflistung aller vorhandenen Ressourcen durch das System bereitgestellt werden (/F60/).

Hintergrundinformationen

Um der Anforderung der Nachvollziehbarkeit und Dokumentation des Assessments nachzukommen, sollte das SISA für das ganze Assessment umfassende Hintergrundinformationen bereitstellen (/Z40/). Informationen sind insbesondere bereitzustellen über die eingesetzte Software (/F70/) und die verwendeten bzw. erzeugten Daten (/F80/) sowie über die verwendeten und erzeugten Dokumenten (/F90/) und die an einer Studie beteiligten Personen und Organisationen (/F100/). Eine Funktion zur Bereitstellung von Informationen über durchgeführte Projekte und Studien (/F110/) sollte ebenfalls vorhanden sein.

Problemverständnis

Das SISA sollte zur Verbesserung der Nachvollziehbarkeit und Transparenz von Ergebnissen auch Hilfestellungen zum Verständnis des Problembereichs liefern (/Z50/). Die Bereitstellung eines Glossars, das die Kernbegriffe des Problemfeldes und des Assessments beinhaltet, wird hier als Minimalanforderung angesehen (/F120/).

Simulation

Eine zentrale Aufgabe eines Systems zum simulationsbasierten integrierten Assessment ist die Erzeugung neuer Simulationsergebnisse (/Z60/). Zu diesem Zweck sollte das System eine Funktion zum Start neuer Simulationsläufe bereitstellen (/F130/). Da die Berechnungen von integrierten Simulationsmodellen sehr zeitaufwendig sein können, sollte das System jederzeit Informationen über deren Verlauf liefern können (/F140/). Diese Informationen sollten z. B. Aufschluss geben über den Fortschritt der Simulationen und über möglicherweise aufgetretene Fehler.

Datenimport

Zur Berechnung neuer Simulationsergebnisse sind umfangreiche Datensätze notwendig. Da die Daten oft aus unterschiedlichen Quellen stammen und keine homogene Datenstruktur aufweisen, ist eine Integration der Daten in das System notwendig. Die Unterstützung des Nutzers bei der Datenintegration wird daher ebenfalls als Ziel des SISA aufgenommen (/Z70/). Zur Erfüllung dieses Zieles ist zumindest eine Funktion zum Import von Datensätzen in das System bzw. zur Konvertierung von Datensätzen in ein bekanntes Systemformat vorzusehen (/F150/).

Datenvorverarbeitung

Aufgrund der Vielfältigkeit der benötigten Daten für ein integriertes Simulationsmodell, müssen einige Datensätze neu erstellt werden. Zur Vorverarbeitung von Daten wird i. d. R. sowohl eigens für diesen Zweck geschriebene Software (Individualsoftware) als auch Standardsoftware eingesetzt (vgl. den Bereich 'Werkzeug' in Abb. 4.1, Seite 73). Die genauen Funktionen zur Datenvorverarbeitung sind projektabhängig und können nicht allgemein angegeben werden. Gleiches gilt für evtl. notwendige Nachbearbeitungen der Ausgabedaten des integrierten Simulationsmodells. Die Unterstützung der Vorverarbeitung und Nachbearbeitung von Daten sollte dennoch als ein Ziel bei der Entwicklung der Architektur berücksichtigt werden (/Z80/) – ihm kann beispielsweise durch eine entsprechende Offenheit gegenüber Standardsoftware oder standardisierten Datenformaten Rechnung getragen werden.

Daten-bereit-stellung

Die vom integrierten Simulationsmodell berechneten Ergebnissen sollen vom SISA bereitgestellt werden (/Z90/). Hierzu soll das System mindestens zwei Funktionen bieten: eine zur Visualisierung der Daten (/F160/) und eine zum Export der Daten in Formate, die von anderen Anwendungen direkt verwendet werden können (/F170/).

Daten-analyse

Die Analyse der über das Simulationsmodell berechneten (und evtl. nachbearbeiteten) Daten soll ebenfalls vom SISA unterstützt werden (/Z100/). Welche Analysen genau anzubieten sind, hängt – wie bei bereits bei der Vorverarbeitung und Nachbearbeitung der Daten des integrierten Simulationsmodells – vom konkret zu realisierenden System ab. Eine Funktion zur statistischen Analyse von Daten (Mittelwert, Median, Standardabweichung etc.) dürfte allerdings für jedes System interessant sein und wird daher als SISA-Funktion gefordert (/F180/).

Modell-analyse

Neben der Berechnung neuer Simulationsergebnisse und der Hilfe bei der Analyse dieser Ergebnisse sollte das SISA auch die Analyse des Simulationsmodells selbst unterstützen (/Z110/). Die genauen Funktionen und Möglichkeiten (beispielsweise zur Durchführung von Sensitivitätsanalysen) hängen stark vom konkret verwendeten Simulationsmodell und dessen Teilmodellen sowie von der gewünschten Ausbaustufe des SISA ab und werden in dieser Arbeit nicht spezifiziert.

Abgren-zungs-kriterien

Nachdem nun die Ziele und Funktionen des SISA angegeben wurden, folgen einige *Abgrenzungskriterien*, die angeben, welche Anforderungen explizit *nicht* an das SISA gestellt werden.

Ziel des SISA ist es *nicht*, ein Simulationswerkzeug bereitzustellen, in dem neue Simulationsmodelle, -modellteile oder Werkzeuge erstellt werden können. Das SISA ist auch *keine* Simulationsumgebung zur dynamischen Auswahl oder Integration von Simulationsmodellen oder -modellteilen. Weiterhin ist das SISA *nicht* dafür zuständig, *alle* Funktionen zur Simulationsergebnis-Analyse bereitzustellen (zur Unterstützung von Analysen sollen die oben genannten Export-Funktionen bereitgestellt werden). Ziel des Systems ist auch *nicht*, als Plattform eingesetzt zu werden, auf die verschiedene Organisationen verteilt zugreifen um an einem Projekt zu arbeiten.

Über-sicht

Eine Übersicht der einzelnen Ziele und Funktionen ist in den Tabellen 4.2 und 4.3 zu finden.

4.2.3 System-Einsatz

Anwen-dungs-bereich

Das SISA wird angewendet im Rahmen von Untersuchungen der Auswirkungen des globalen Wandels, insbesondere im Bereich der Klimafolgenforschung. Die aufbereiteten Ergebnisse des Systems sollen politischen Entscheidungsträgern als Informationsgrundlage dienen und darüber hinaus zur Identifizierung offener Fragestellungen innerhalb der beteiligten wissenschaftlichen Fachdisziplinen beitragen.

Name des Ziels	Kennzeichnung
Verwaltung von Simulationsläufen	/Z10/
Verwaltung von Assessment-Ergebnissen	/Z20/
Verwaltung von Ressourcen	/Z30/
Bereitstellung von Hintergrundinformationen	/Z40/
Förderung des Problemverständnisses	/Z50/
Erzeugung neuer Simulationsergebnisse	/Z60/
Integration von Daten	/Z70/
Unterstützung bei der Vorverarbeitung und Nachbearbeitung von Simulationsmodell-Daten	/Z80/
Bereitstellung von Daten	/Z90/
Unterstützung der Datenanalyse	/Z100/
Unterstützung der Modellanalyse	/Z110/

Tabelle 4.2: Übersicht der grundlegenden Ziele des Systems zum simulationsbasierten integrierten Assessment. Erklärungen zu den Zielen finden sich im Unterabschnitt 4.2.2 (Seite 78).

Name der Funktion	Kennzeichnung	Ziel
Beschreibung eines Szenarios	/F10/	/Z10/
Zuordnung von Daten zu einem Szenario	/F20/	/Z10/
Beschreibung eines Simulationslaufes	/F35/	/Z10/
Zuordnung von Daten zu einem Simulationslauf	/F30/	/Z10/
Bereitstellung von Metadaten	/F40/	/Z10/
Zuordnung von Ressourcen zu Projekten	/F50/	/Z20/
Bereitstellung einer Ressourcen-Liste	/F60/	/Z30/
Bereitstellung von Assessment-bezogenen Informationen zu: - verwendeter Software - verwendeten/erzeugten Daten - verwendeten/erzeugten Dokumenten - beteiligten Personen/Organisationen - durchgeführten Projekten/Studien	 /F70/ /F80/ /F90/ /F100/ /F110/	/Z40/
Bereitstellung eines Glossars	/F120/	/Z50/
Start eines Simulationslaufes	/F130/	/Z60/
Bereitstellung von Informationen zum Simulationsverlauf	/F140/	/Z60/
Import/Konvertierung von Datensätzen	/F150/	/Z70/
Visualisierung von Daten	/F160/	/Z90/
Export von Daten	/F170/	/Z90/
Statistische Auswertung von Daten	/F180/	/Z100/

Tabelle 4.3: Übersicht der grundlegenden Funktionen des Systems zum simulationsbasierten integrierten Assessment. Erklärungen zu den Zielen finden sich im Unterabschnitt 4.2.2 (Seite 78).

Ziel-gruppen

Betrieben wird das SISA von Wissenschaftlern; sie erzeugen szenarienbezogene Simulationsergebnisse, werten diese aus und vermitteln die Ergebnisse den Entscheidungsträgern und anderen Wissenschaftlern.[8] Entscheidungsträger und andere interessierte Personen oder Personengruppen sollen das System nutzen können, um sowohl die Ergebnisse von Studien als auch Hintergrundinformationen zu den Berechnungen direkt abrufen zu können.

Eine Übersicht der vorgesehenen Nutzung des SISA ist in Abb. 4.3 in Form eines Anwendungsfalldiagramms zu finden.

Betriebs-bedin-gungen

Das System soll in der Umgebung wissenschaftlicher Einrichtungen eingesetzt werden; hier werden neue Simulationsergebnisse erzeugt, Analysen erstellt und interessierten Personengruppen bereitgestellt.

Der Einsatz des SISA zur Berechnung neuer Simulationsergebnisse macht nur einen geringen Anteil an der Gesamteinsatzzeit des Software-Systems aus. Während dieser Zeit findet eine intensive Rechnernutzung durch die Simulationsmodelle statt. Die anderen Funktionen des SISA verursachen eine geringe Rechnerlast und können im Hintergrund ablaufen.

4.2.4 System-Umgebung

Das SISA sollte auf einem Arbeitsplatzrechner ablauffähig sein. Die Verteilung des Systems auf mehrere Rechner (z. B. die Auslagerung einer Datenbank) ist wünschenswert.

Software

Die Software-Umgebung setzt sich zusammen aus den standardmäßig verwendeten Büroprogrammen (Textverarbeitung, Tabellenkalkulation), einem Internet-Browser, einem Datenbank-Managementsystem (DBMS) und einem Geo-Informationssystem (GIS). Zur Datenanalyse werden z. B. Statistikprogramme eingesetzt. Der Einsatz des SISA sollte sowohl unter den aktuellen Betriebssystemen der Firma *Microsoft* möglich sein als auch unter dem Betriebssystems *Linux*.

Schnitt-stellen

Schnittstellen zur oben angeführten Software-Umgebung sollten bereitgestellt werden. Insbesondere Schnittstellen zum DBMS und zu den Daten und Funktionen des GIS sind, falls diese Systeme eingesetzt werden, notwendig.

Hard-ware

Die Anforderungen an die Hardware hängen vom konkret zu realisierenden SISA ab und reichen von standardmäßig ausgerüsteten Personal Computern (PC) bis hin zu sehr leistungsstarken Rechnern.

[8]Die Vermittlung von Ergebnissen kann nach Tol und Vellinga (1998) auf zwei grundsätzlich verschiedenen Wegen geschehen; erstens über die direkte Kommunikation (Präsentation der Analysen und Ergebnisse für die Entscheidungsträger), zweitens über eine indirekte Kommunikation (Ergebnisse werden über die wissenschaftliche Literatur verbreitet und von unabhängigen Kommentatoren oder Beratern zusammengefasst und in ein für Entscheidungsträger angepasstes Format gebracht). Zu den Vor- und Nachteilen beider Wege siehe Tol und Vellinga (1998).

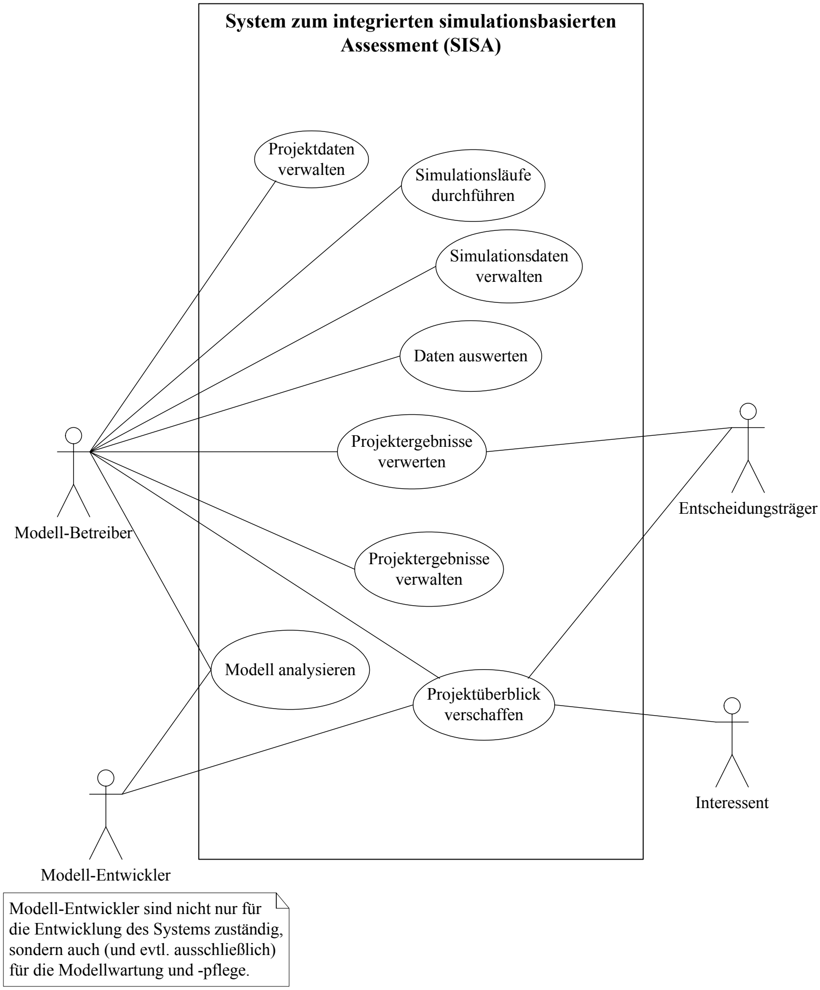


Abbildung 4.3: Anwendungsfalldiagramm des SISA. Zu sehen sind die einzelnen Nutzer des Systems (die Akteure) und die einzelnen Anwendungsfälle, für die das System Funktionen bereitzustellen hat. Der *Modell-Betreiber* nutzt das SISA zur Verwaltung von Projektdaten, Simulationsdaten und Projektergebnissen sowie zur Durchführung von Simulationsläufen und der Analyse der vorhandenen Daten. Die Projektergebnisse werden dem Modell-Betreiber und dem *Entscheidungsträger* vom System zur Verwertung bereitgestellt. Die Funktionen zur Analyse des Modells werden vom Modell-Betreiber und/oder vom *Modell-Entwickler* verwendet. Allgemeine (Hintergrund-)Informationen sind allen Akteuren (inkl. allgemeiner *'Interessenten'*) vom System bereitzustellen. Ausführlichere Angaben zu den Zielen und Funktionen finden sich im Text.

Orgware

Das SISA muss organisatorisch in den Betrieb des Modellbetreibers eingebettet werden. Folgende organisatorische Randbedingungen (Orgware) werden gesetzt: Die Verwaltung und Pflege der benötigten Daten und der evtl. vorhandenen Datenbank wird von fachkundigen Mitarbeitern übernommen. Zur Integration neuer Simulationsmodelle oder -modellteile stehen Mitarbeiter zur Verfügung, die sich mit der Software-Architektur des Systems auskennen und notwendige Änderungsarbeiten anleiten oder selber durchführen.

4.2.5 System-Daten

Einteilung

Zur Erfüllung der spezifizierten Ziele und Funktionen sind vom SISA unterschiedlich detaillierte Informationen (Dokumente, Daten, Metadaten) vorzuhalten. Bei den zu speichernden Informationen kann unterschieden werden zwischen den Daten und Metadaten, die vom Nutzer für die Verwaltung der Ressourcen und des Assessments selbst sowie zum Verständnis von Studien benötigt werden (Hintergrunddaten), den Daten, die als Primär-, Sekundär- und Simulationsmodell-Daten benötigt werden (Assessment-Daten) und Daten, die zur Konfiguration von Simulationsläufen notwendig sind (Konfigurationsdaten).

Hintergrunddaten

Zu den Hintergrunddaten gehören Informationen über Projekte, über beteiligte Personen und deren Organisationszugehörigkeit sowie über verwendete Szenarien und durchgeführte Simulationsstudien und Simulationsläufe (vgl. Abb. 4.4). Eine Liste mit vorhandenen Ressourcen und die Speicherung von Metadaten zu den Ressourcen wird ebenfalls zur Erfüllung der Funktionen und Ziele aus Unterabschnitt 4.2.2 benötigt. Die Daten zum Glossar fallen ebenfalls in die Kategorie der Hintergrunddaten. Für Simulationsstudien, Simulationsläufe, Szenarien und Projekte sind detaillierte Beschreibungen in Form von Dokumenten zu erstellen. Da alle Dokumente zu den SISA-Ressourcen gehören (vgl. Abb. 4.2, Seite 76), sind diese auch in der Ressourcenliste aufzuführen und über Metadaten zu beschreiben.

Dokumente

Eine detaillierte Spezifikation der Hintergrunddaten erfolgt im Rahmen der Architekturentwicklung (Kapitel 5, Seite 95). Zusammenfassend kann aber festgehalten werden, dass das SISA die folgenden Hintergrunddaten zur Verfügung zu stellen hat (Referenzierungen erfolgen über die Kurzform /Dxx/): Kurzinformationen über Personen und Organisationen (/D10/), Kurzinformationen über die abstrakten Ressourcen (/D20/), Metadaten für die Ressourcen (/D30/) und Glossar-Einträge (/D40/).

Konfigurationsdaten

Informationen über die Zuordnung von Datensätzen zu den einzelnen Simulationsläufen/Szenarien bilden die Konfigurationsdaten (/D50/) des Systems.

Assessment-daten

In die Gruppe der Assessment-Daten gehören alle weiteren, in Abb. 4.2 (Seite 76) aufgeführten Datensätze (mit Ausnahme der Systemparameter- und Optionsdaten, die für die Konfiguration zuständig sind). Die Auswahl wichtiger Assessment-Daten erfolgt in den folgenden Absätzen.

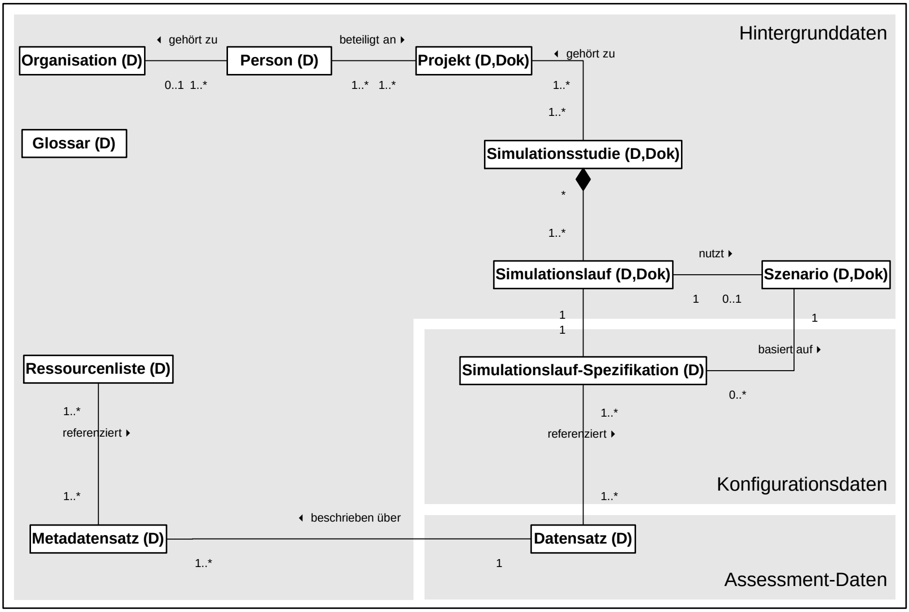


Abbildung 4.4: Anforderung an die System-Daten. Für die dargestellten Klassen müssen zur Erfüllung der Systemziele und -funktionen Daten innerhalb des SISA verwaltet werden (Kennzeichnung 'D'). Die Attribute der einzelnen Klassen werden im Kapitel der Architekturentwicklung (Kap. 5, Seite 95) spezifiziert. Für einige Klassen werden i. d. R. detaillierte Beschreibungen in Form von Dokumenten erzeugt (Kennzeichnung 'Dok'). Die Metadaten dieser Dokumente sind – genauso wie die Metadaten zu allen anderen Ressourcen – über die Ressourcenliste zu integrieren (vgl. auch Abb. 4.1, Seite 73 und Abb. 4.2, Seite 76).

Assessment-Daten

Die Assessment-Daten sind abhängig von konkreten Projekten. Eine Auswahl von grundlegenden Daten, die bei integrierten Assessments im Rahmen von Fragestellungen des globalen Wandels häufig benötigt werden, soll in diesem Unterabschnitt dennoch getroffen werden.

Überblick

Grundlegende Daten des SISA sind diejenigen Daten, die die Grundcharakteristiken der einzelnen, wichtigen Komponenten der Natur- und Anthroposphäre beschreiben (zu den einzelnen Komponenten siehe Kapitel 2.1.1, Seite 7). Aus dem Bereich der *Hydrosphäre* sind Daten über Fluss-Einzugsgebiete, Flüsse, Flussrichtungen, Seen und anderen Feuchtgebieten notwendig. Innerhalb der *Biosphären-Komponente* ist die Landnutzung und die Landbedeckung (die natürliche Vegetation) von Interesse. In den Bereich der *Lithosphäre/Pedosphäre* fallen Angabe über Bodentypen. Die wichtigsten Informationen aus der *Bevölkerungs-Komponente* sind die Gesamtbevölkerung und die Bevölkerungsstruktur. Wichtige Daten zum Thema *Verkehr* sind Angaben über vorhandene Transportwege (Straßen, Schienen, Flüsse, Kanäle). Im *Wirtschaftsbereich* spielt das Brutto-Sozialprodukt eine ausgesprochen wichtige Rolle beim integrierten Assessment. Darüber hinaus ist die Einkommensverteilung ein wichtiger Indikator bei einigen Auswirkungsanalysen. Aus dem Bereich der *Wissenschaft/Technik* sind grundlegende Angaben über den Stand der Agrartechnologie von Interesse (z. B. was die Düngernutzung oder die Bewässerung landwirtschaftlicher Flächen angeht). Daten aus dem Bereich der *gesellschaftlichen Organisation* beschreiben beispielsweise (Wirtschafts-)Regionen – auch die i. d. R. benötigten Ländergrenzen können hier eingeordnet werden. Die Sammlung relevanter Daten aus der *psychosozialen Sphäre* ist – ebenso wie die Modellierung der zugehörigen Prozesse – noch nicht so weit fortgeschritten wie diejenige aus den anderen Komponenten, so dass für diesen Bereich keine grundlegenden Datensätze angegeben werden.[9]

HYDE

Das staatliche Institut für Gesundheit und Umwelt der Niederlande (RIVM[10]) hat zur Validierung von Simulationsmodellen eine Umfangreiche Datenbank erstellt; die *Hundred Year Database for Integrated Environmental Assessments (HYDE)* (Klein Goldewijk und Battjes, 1997). HYDE enthält Daten zu folgenden Themen: 1) *Grundlegende Triebkräfte* (Bevölkerung, Bruttosozialprodukt, Wertschöpfung durch Industrie und Dienstleistungen, Privat-Konsum, Anzahl PKW, Temperatur und Niederschlag), 2) *Ökonomie des Energie- und Industriesektors* (beispielsweise Energieverbrauch, Elektrizitätserzeugung, Brennstoffpreise, Produktion von Kohle, Öl, Eisen und anderen Metallen), 3) *Emissionen des Energie- und Industriesektors* (beispielsweise für

[9] Zu Daten aus der psychosozialen Sphäre können z. B. Beschreibungen der *Wahrnehmung* der Umwelt und Beschreibung von Handlungszielen und -mitteln gezählt werden.

[10] RIVM = Rijksinstituut voor Volksgezondheid en Milieu, engl. National Institute of Public Health and the Environment.

Kohlendioxid, Methan und Schwefeldioxid), 4) *Terrestrische Umwelt* (beispielsweise historische Landbedeckung, Nahrungs- und Futtermittelverbrauch, Feldfruchtproduktion, Anzahl Tiere, Handel landwirtschaftlicher Produkte, Düngerverbrauch), 5) *Atmosphäre und Ozean* (beispielsweise Konzentrationen von Kohlendioxid und anderen klimarelevanten Gasen, Landbedeckung, Ozean-Temperatur und Eisbedeckung des Meeres).

Da nicht alle angeführten Daten für jedes Assessment benötigt werden, werden im Folgenden einige wichtige Datensätze ausgewählt, die als Grundlage für jedes SISA dienen sollen.

Auswahl der Assessment-Daten

Modelle zum integrierten Assessment sind oft nach dem so genannten DPSIR-Prinzip aufgebaut: die Abbildung der Wirkungskette eines betrachteten Problems beginnt mit den treibenden Faktoren (Driving forces), die dann zu einem bestimmten Druck (Pressure) auf das System und einem bestimmten Zustand (State) innerhalb des Systems führen. Dieser Zustand führt dann zu verschiedenen Auswirkungen (Impacts), auf die mit bestimmten Antworten (Responses) reagiert wird. Näheres zu diesem Prinzip findet sich z. B. in Peirce (1998) und Luiten (1999).

Grundlagen

Um die Konsistenz und Vergleichbarkeit der Ergebnisse unterschiedlicher Studien und Projekte zu ermöglichen, sollten die treibenden Faktoren, d. h. die exogenen Einflussfaktoren bzw. Modellumweltdaten, möglichst unabhängig vom konkreten Projekt sein. Zu den grundlegenden Daten, die für viele Assessments benötigt werden, gehören Angaben über die ökonomischen und demographischen Zustände und Entwicklungen von Regionen. Daten über die Einwohnerzahl von Regionen (/D60/) und das oft als Indikator benutzte Brutto-Sozialprodukt pro Einwohner (/D70/) sollten daher zu den grundlegenden Assessment-Daten eines SISA gehören. Eine Liste vorhandener Regionen[11] (/D80/) muss ebenso vorhanden sein, wie Geo-Datensätze, die die Regionen voneinander abgrenzen: ein hoch aufgelöster (Vektor-)Datensatz zur Beschreibung der Regionengrenzen (/D90/) ist ebenso notwendig wie ein (Raster-)Datensatz (/D100/), der die Regionen in der Fläche darstellt. Mit der Darstellung der Regionenfläche verbunden, ist ein weiterer grundlegender Datensatz: die Landmaske (/D110/). Die Landmaske, die abhängig ist von der geographischen Auflösung des Assessments, legt fest, für welche Teile der Erde landbezogene Berechnungen durchzuführen sind. Untersuchungen im Rahmen des globalen Wandels benötigen darüber hinaus i. d. R. Klimadaten (/D120/).

Die aufgeführten Daten stellen – wie bereits angemerkt – lediglich eine Grundausstattung dar. Die tatsächlich zu speichernden Daten sind abhängig

[11] Eine Region kann sowohl ein Land als auch ein Teil eines Landes oder eine Zusammensetzung mehrerer Länder sein.

vom konkreten Systemziel des SISA. Eine Übersicht der vorzuhaltenden Daten ist in Tab. 4.4 zu finden.

Bezeichnung der Datensätze	Kennzeichnung
Kurzinformationen über Personen/Organisationen	/D10/
Kurzinformationen über 'abstrakte Ressourcen'	/D20/
Metadaten zu Ressourcen	/D30/
Glossar-Einträge	/D40/
Konfigurationsdaten	/D50/
grundlegende Assessment-Daten	/D60-D120/

Tabelle 4.4: Übersicht der grundlegenden Datenbestände des Systems zum simulationsbasierten integrierten Assessment. Erklärungen zu den Datensätzen finden sich im Text.

4.2.6 System-Leistungen

zeitliche Anforderung

Es es nicht vorgesehen, dass Entscheidungsträger interaktiv neue Simulationsergebnisse erzeugen. Simulationsläufe werden hingegen von den modellbetreibenden Wissenschaftlern (Modell-Betreibern) erzeugt. Aufgrund der Wichtigkeit von Unsicherheits- und Sensitivitätsanalysen und der damit verbundenen Notwendigkeit mehrere Simulationsläufe mit sich ändernden Modell-Eingangsgrößen durchzuführen, sollte sich die Berechnung neuer Simulationsergebnisse in einem akzeptablen, vom konkreten SISA abhängigen Rahmen halten. Eine generelle Aussage über die Performanz des Systems kann an dieser Stelle allerdings nicht getroffen werden.

Simulationsdauer

Eine Leistungsanforderung wird allerdings als Wunschanforderung angegeben: Für einen gestarteten Simulationslauf sollte noch am gleichen Tag geprüft werden können, ob die Berechnungen verwertbar sind oder nicht. Berücksichtigt man eine gewisse Arbeitszeit zur Vorbereitung und Nachbearbeitung einer Simulation, so kann die folgende Leistungsanforderung formuliert werden: Die Dauer eines Simulationslaufes sollte sechs Stunden nicht überschreiten (Leistungsanforderung /L10/).

Datenauflösung u. -abdeckung

Der Umfang und die Genauigkeit der zu speichernden Daten hängt ebenfalls von der konkreten Anwendung des SISA ab. Ein Charakteristikum von integrierten Assessments im Rahmen des globalen Wandels ist die geographisch explizite Abbildung von Prozessen. Die Produktdaten müssen daher auch einen definierten geographischen Bereich abdecken. Im Rahmen der Klimafolgenforschung umspannt dieser Bereich i. d. R. die gesamte Erde. Die geographische Auflösung variiert anwendungsspezifisch, beläuft sich für viele flächenbezogene Datensätze aber auf 0.5 Grad mal 0.5 Grad geographischer Länge/Breite

(vgl. z. B. Alcamo u. a., 1998a, Döll u. a., 2003; Klein Goldewijk, 2001; Klein Goldewijk und Battjes, 1997, New u. a., 1999; New u. a., 2000). Regionenbezogene Daten beziehen sich auf Länder oder Zusammenfassungen von Ländern. Die Anzahl der Regionen variiert von etwa einem Dutzend bis zu über 200 (vgl. Dowlatabadi, 1995; Weyant u. a., 1996). Die zeitliche Auflösung der Daten variiert ebenfalls von Anwendung zu Anwendung und ist oft durch die zur Verfügung stehenden Daten begrenzt (s. o. g. Zitate). Für die grundlegenden Daten kann als Leistungsanforderung lediglich definiert werden, dass die Daten in einer den konkreten Anforderungen genügenden geographischen und zeitlichen Auflösung und Abdeckung vorzuliegen haben.

4.2.7 Benutzungsschnittstellen

Dieser Abschnitt legt die Anforderungen an die Benutzungsschnittstelle (referenziert über /Bxx/) fest, die sowohl die Schnittstelle zu den einzelnen Akteuren, als auch die Schnittstelle zu anderen Software-Systemen beinhaltet.

Aus Gründen der einfachen Bedienbarkeit sollte die Nutzung des SISA über einen Internet-Browser möglich sein (/B10/).

Zur Arbeit mit dem SISA sind den Akteuren (Modellbetreiber, Modellentwickler, Entscheidungsträger) verschiedene ‘Sichten’ auf die Daten und Leistungen bereitzustellen (/B20/). Die Sichten und die mit ihnen verbundenen Möglichkeiten des Datenzugriffs und der Funktionsausführung ergeben sich aus dem Anwendungsfalldiagramm (Abb. 4.3, Seite 83). Modellentwickler und Modellbetreiber dürfen auf alle gespeicherten Daten zugreifen. Entscheidungsträgern sollte der Zugriff auf die Analyseergebnisse und auf Hintergrundinformationen erlaubt werden. Außenstehenden (Interessenten) sollte der Zugriff auf allgemein beschreibende Daten und Ansprechpartner gestattet werden.

Anderen Software-Systemen sollte eine Programmierschnittstelle (Application Programming Interface, API) zur Verfügung gestellt werden, die es erlaubt, einen Simulationslauf zu starten und bereits berechnete Ergebnisse abzufragen (/B30/).

4.2.8 Qualitäts-Zielbestimmung

Die anzustrebende Qualität des SISA wird über die Merkmale und Teilmerkmale der ISO/IEC 9126 (vgl. Tabelle 2.1, Seite 23) angegeben. Eine Übersicht der nachfolgend beschriebenen Qualitätsziele liefert Tabelle 4.5 (Seite 90).

Das Qualitätsmerkmal der *Funktionalität* umfasst fünf Teilmerkmale: Angemessenheit, Richtigkeit, Interoperabilität, Ordnungsmäßigkeit und Sicherheit. Funktionalität
Die Anforderungen an die *Angemessenheit* und die *Sicherheit* können als *normal* eingestuft werden (sie sind relevante Merkmale, aber es werden an sie keine besonderen oder außergewöhnlichen Anforderungen gestellt). Aufgrund

Merkmal	Teilmerkmal	Produktqualität			
		sehr gut	gut	normal	nicht rel.
Funktionalität	Angemessenheit			X	
	Richtigkeit		X		
	Interoperabilität	X			
	Ordnungsmäßigkeit		X		
	Sicherheit			X	
Zuverlässigkeit	Reife			X	
	Fehlertoleranz			X	
	Wiederherstellbarkeit			X	
Benutzbarkeit	Verständlichkeit			X	
	Erlernbarkeit			X	
	Bedienbarkeit			X	
Effizienz	Zeitverhalten		X		
	Verbrauchsverhalten		X		
Änderbarkeit	Analysierbarkeit	X			
	Modifizierbarkeit	X			
	Stabilität			X	
	Prüfbarkeit		X		
Übertragbarkeit	Anpassbarkeit		X		
	Installierbarkeit		X		
	Konformität		X		
	Austauschbarkeit	X			

Tabelle 4.5: Anforderungen des SISA an die Software-Qualität. Qualitätsmerkmale angelehnt an ISO/IEC 9126 (DIN 66272). Erklärungen zu den Merkmalen finden sich im Text und in Tabelle 2.1 (Seite 23).

der Komplexität der Simulationsmodelle und der damit nicht einfach nachzuvollziehende Ergebnisberechnungen und Ergebnisse ist an die *Richtigkeit*, insbesondere der Ausgaben des Simulationsmodells und seiner Teile, eine *hohe Anforderung* zu stellen (ein Modellteil ist oftmals nur von einem Fachexperten – der i. d. R. auch selbst der Autor des Programms ist – nachvollziehbar). Die *Interoperabilität*, also die Eignung, mit anderen Systemen zusammenzuwirken, spielt aufgrund der Komplexität und der Umgebungsbedingungen bei der Erstellung und dem Einsatz des Gesamtsystems eine wichtige Rolle. Die Produktqualität sollte diesbezüglich daher *sehr gut* sein. Das Teilmerkmal der *Ordnungsmäßigkeit* beschreibt die Erfüllung anwendungsspezifischer Normen und Vereinbarungen. Auch wenn es für die in Kapitel 72 (Seite 87) angeführten wichtigsten Daten des SISA keine Bestimmungen gibt, sollte bei der Auswahl der Produktdaten darauf geachtet werden, dass es sich um allgemein akzeptierte Daten handelt. Hierzu zählen z. B. Angaben der Weltbank und der Vereinten Nationen oder ihrer Unterorganisationen (wie etwa der Welternährungs- und Landwirtschaftsorganisation (FAO) oder der Weltgesundheitsorganisation (WHO)). Das Qualitätsziel in Bezug auf das Teilmerkmal der *Ordnungsmäßigkeit* wird aufgrund dieser Anforderung auf *gut* gesetzt.

Zuverlässigkeit u. Benutzbarkeit

Die Qualitätsanforderungen hinsichtlich der Zuverlässigkeit des SISA, die über die Teilmerkmale *Reife*, *Fehlertoleranz* und *Wiederherstellbarkeit* ausgedrückt werden, werden als *normal* eingestuft. Die Qualitätsanforderungen an die Benutzbarkeit mit ihren Teilmerkmalen *Verständlichkeit*, *Erlernbarkeit* und *Bedienbarkeit* werden ebenso als *normal* festgelegt.

Effizienz

Das Qualitätsmerkmal der *Effizienz* dient der Beurteilung des Verhältnisses von erreichter Leistung zu eingesetzten Betriebsmitteln. Das SISA stellt an beide Teilmerkmale der Effizienz, das *Zeitverhalten* und das *Verbrauchsverhalten*, besondere Anforderungen: Die Durchführung von Simulationen, also die Berechnung neuer Modellergebnisse mit Hilfe des Simulationsmodells, fordert sehr komplexe Berechnungen. Das *Zeitverhalten* spielt hier eine besondere Rolle – und wurde daher auch als explizite Leistungsanforderung (/L10/) im Kapitel 4.2.6 (Seite 88) aufgeführt. Die geforderte Qualität des SISA (insbesondere des Simulationsmodells) hinsichtlich des *Zeitverhaltens* wird daher auf *gut* gesetzt.

Das Teilmerkmal des *Verbrauchsverhaltens* dient der Beschreibung des Aufwandes an Betriebsmitteln. Diese Betriebsmittel können sowohl andere Software-Produkte, Hardware-Einrichtungen und Materialien (wie Druckerpapier oder Disketten) als auch Dienstleistungen von Bedienungs-, Wartungs-, oder Unterstützungspersonal einschließen (DIN, 1994). An dieser Stelle ist eine Abwägung zu machen zwischen dem Verbrauchsverhalten und der Erreichung der anderen qualitativen und funktionalen Ziele: So ist beispielsweise der Einsatz eines Datenbankmanagementsystems (DBMS) hilfreich bei der Verwaltung von Projektdaten und der Auswertung von Simulationsergebnissen; zur Pflege DBMS bedarf es aber eines erhöhten Personaleinsatzes. Das Zeitverhalten des

Systems kann, als zweites Beispiel, durch den Einsatz einer besseren Hardware-Ausrüstung erreicht werden; die beiden Teilmerkmale der Effizienz stehen sich hier also direkt gegenüber. Da das SISA in der Umgebung wissenschaftlicher Einrichtungen eingesetzt wird (siehe Kapitel 4.2.4, Seite 82) ist aufgrund der oft vorherrschenden angespannten Ressourcenlagen eine *gute* Qualität bezüglich des *Verbrauchsverhaltens* an das Produkt zu stellen.

Änderbarkeit

Das SISA integriert unterschiedliche Daten und Teilmodelle. Während der Einsatzzeit des SISA sollten die verwendeten Daten und Simulationsmodelle stets den neuesten Stand der wissenschaftlichen Erkenntnisse widerspiegeln. Diese Anforderung bedingt die Notwendig ständiger Korrekturen, Verbesserungen und Anpassungen des Software-Systems. Das entsprechende Qualitätsmerkmal innerhalb der ISO/IEC 9126 ist die *Änderbarkeit* mit ihren Teilmerkmalen Analysierbarkeit, Modifizierbarkeit, Stabilität und Prüfbarkeit.

Änderungen am SISA sind oft von Personen durchzuführen, die nicht an der Entwicklung des Gesamtkonzeptes oder eines bestimmten Teilmodells des SISA mitgearbeitet haben. Die *Analysierbarkeit* des Systems, die den notwendigen Aufwand zur Identifizierung änderungsbedürftiger Teile einschließt, muss daher *sehr gut* sein. Hinsichtlich der *Modifizierbarkeit* des Systems wird, analog zu den vorhergehenden Überlegungen, ebenfalls eine *sehr gute* Qualität gefordert. An die *Stabilität* des Gesamtsystems (Risiko unerwarteter Wirkungen) im Zusammenhang mit durchgeführten Änderungen werden *keine außergewöhnlichen Anforderungen* gestellt. Eine *gute Prüfbarkeit* des geänderten Software-Systems wird allerdings als Qualitätsziel aufgenommen.

Übertragbarkeit

SISA und SISA-Teile (insbesondere das Simulationsmodell bzw. Teile des Simulationsmodells) werden zunehmend zwischen verschiedenen Organisationen ausgetauscht. Die Systemumgebungen können dabei nicht als homogen angenommen werden: Es werden unterschiedliche DBMS, GIS und andere Software-Produkte eingesetzt, und auch die Betriebssysteme können unterschiedlich sein. Darüber hinaus ist die Systemumgebung innerhalb einer Organisation auch nicht stabil. Die *Übertragbarkeit* des SISA von einer Umgebung in eine andere spielt daher eine Rolle bei dessen Entwicklung. Die *Anpassbarkeit* des Systems an neue Umgebung und die *Installierbarkeit* innerhalb einer vorgegebenen Umgebung sollte daher *gut* sein. Die Einhaltung von Normen führt zu einer besseren Übertragbarkeit – dies gilt sowohl für Teile des SISA als auch für die erzeugten und verwendeten Daten. Das System sollte daher eine *gute Konformität* mit solchen Normen aufweisen. Um den bereits oben angesprochenen, wichtigen Austausch von Modellteilen des Simulationsmodells mit anderen Organisationen zu ermöglichen und zu erleichtern, sollte bei der Entwicklung des Software-Systems eine *sehr gute Austauschbarkeit* (zumindest von Modell-Teilen) explizit angestrebt werden.

Aufgrund der in Unterabschnitt 4.2.1 (Seite 75) angeführten Wichtigkeit einer guten Dokumentation wird die Erfüllung einer weitere nicht-funktionale

Anforderung als *sehr wichtig* eingestuft: die der *Transparenz, Nachvollziehbarkeit und Reproduzierbarkeit* von Assessment-Ergebnissen.

Die vorrangig zu berücksichtigenden nicht-funktionalen Anforderungen sind in Tabelle 4.6 noch einmal zusammengefasst und mit referenzierbaren Kennzeichnungen (/NFxx/) versehen.

Name des nicht-funktionalen Anforderung	Kennzeichnung
Interoperabilität	/NF10/
Analysierbarkeit	/NF20/
Modifizierbarkeit	/NF30/
Austauschbarkeit	/NF40/
Transparenz, Nachvollziehbarkeit, Reproduzierbarkeit	/NF50/

Tabelle 4.6: Übersicht der vorrangigen nicht-funktionalen Anforderungen an das SISA. Erklärungen zu den nicht-funktionalen Anforderungen finden sich im Text. Eine ausführlichere Auflistung zur Wichtigkeit der einzelnen nicht-funktionalen Anforderungen findet sich in Tabelle 4.5 (Seite 90).

4.2.9 Testszenarien

In Anlehnung an die Zielbestimmungen in Kapitel 4.2.2 (Seite 78) sind zum Test eines SISA die Funktionen zu den folgenden, als vorrangig betrachteten Zielen zu prüfen: die Verwaltung von Simulationsläufen (Testszenario /T10/), Assessment-Ergebnissen (/T20/) und Ressourcen (/T30/), die Bereitstellung von Hintergrundinformationen (/T40/), die Erzeugung neuer Simulationsergebnisse (/T60/), die Integration von Daten in das System (/T70/) und die Datenbereitstellung (/T90/).

4.2.10 Entwicklungs-Umgebung

Die Umgebung für die Entwicklung des SISA ist abhängig von der konkreten Ausprägung der im Kapitel 4.2.4 (Seite 82) skizzierten System-Umgebung (z. B. dem tatsächlich eingesetzten GIS oder einem vorhanden DBMS). Bei der Auswahl der Software-Entwicklungsumgebung ist auf die geforderte Übertragbarkeit des Systems auf andere Umgebungen zu achten. Programmiersprachen sollten aus diesem Grund nur in einem Umfang verwendet werden der standardisiert ist; die Verwendung betriebssystemspezifischer Software-Komponenten (wie der *Microsoft Foundation Classes*) ist zu vermeiden.

4.3 Fazit

Ziel

Ziel des Systems zum integrierten simulationsbasierten Assessment (SISA) ist die Unterstützung des integrierten Assessments durch die Bereitstellung eines konsistenten Rahmens für Daten und Simulationsmodelle zum System Erde und zur Durchführung von Simulationsläufen sowie die Bereitstellung grundlegender Informationen zu durchgeführten oder in der Durchführung begriffenen Projekten.

Ressourcen

Um dieses Ziel zu erreichen, müssen verschiedene Betriebsmittel (Ressourcen) durch das SISA verwaltet werden. Zu diesen Ressourcen gehören sowohl die Simulationsmodelle und die ihnen zugeordneten Daten als auch andere Software (z. B. zur Vorverarbeitung oder Nachbearbeitung von Daten) und Dokumente (z. B. Modellbeschreibungen oder Ergebnisberichte). Darüber hinaus muss das SISA Informationen über Projekte, Analysen, Szenarien, beteiligte Personen und andere Hintergrundinformationen bereitstellen. Die Daten, die für die Simulationsmodelle benötigt werden, sollten ebenfalls über das SISA zur Verfügung stehen. Die Durchführung und Verwaltung von Simulationsläufen und die Bereitstellung der Simulationsergebnisse gehören darüber hinaus ebenso zur Aufgabe des SISA wie die Bereitstellung von Simulationsergebnissen. Die Sicherstellung der Konsistenz wird unterstützt durch die Dokumentation der Simulationsergebnisse, der verwendeten Simulationsmodelle, der zugrunde liegenden Simulationslauf-Spezifikation und des Simulationslaufes selbst.

Daten

Simulationen

Konsistenz

Umgebung

Das SISA sollte in eine Software-Umgebung eingebettet werden können. Insbesondere zu Geo-Informationssystemen (GIS) und zu Datenbank-Managementsystemen (DBMS) sollten Schnittstellen vorhanden sein.

Qualität

Neben der gewünschten *Interoperabilität* mit GIS und DBMS stellt das SISA weitere Anforderungen an die Qualität der Software-Architektur: Wegen des zunehmend notwendigen Austausches von Modellteilen zwischen unterschiedlichen Organisationen sollte das Qualitätsmerkmal der *Austauschbarkeit* (zumindest von Modellteilen) bei der Entwicklung der Architektur besonders berücksichtigt werden. Die *Modifizierbarkeit* des Systems (insbesondere von Modellteilen) sollte wegen der oft notwendigen Änderungen und Aktualisierungen ebenfalls sehr gut sein. Darüber hinaus wird die *Transparenz, Nachvollziehbarkeit und Reproduzierbarkeit* von Assessment-Ergebnissen als wichtiges Qualitäts-Merkmal eines SISA definiert. Um die Anforderung der *Nachvollziehbarkeit* von Assessment-Ergebnissen zu erfüllen, ist die Erfüllung des Qualitätsmerkmals der *Analysierbarkeit* ebenfalls besonders zu berücksichtigen.

Nachdem die Ziele und Funktionen des SISA in diesem Kapitel spezifiziert wurden, erfolgt im nächsten Kapitel die Entwicklung einer Software-Architektur für ein solches System.

Kapitel 5

Architektur-Entwicklung

In Kapitel 4 wurden allgemeine, *projektunabhängige* Anforderungen an ein System zum integrierten simulationsbasierten Assessment definiert. Um die nichtfunktionalen (qualitativen) Anforderungen des Systems (wie die Modifizierbarkeit, Austauschbarkeit und Interoperabilität) zu erfüllen, muss das System in seine *grundlegenden Bestandteile* gegliedert werden. Die Einteilung des Gesamtsystems in seine Bestandteile (Komponenten) ergibt, in Verbindung mit einer Definition der *'Verantwortlichkeiten'* jeder Komponente, die *Software-Architektur* des Systems. Architektur

Die folgenden Abschnitte beschäftigen sich mit der *Abgrenzung* der einzelnen Komponenten sowie der Definition der *Verantwortlichkeiten*, *Schnittstellen* und grundlegenden *Datenstrukturen*. Die Gesamtfunktionalität des Systems ergibt sich aus der Interaktion der einzelnen Komponenten. Durch die Schnittstellen werden die Operationen beschrieben, mit denen der Datenaustausch zwischen den Komponenten koordiniert wird. Die grundlegenden Datenstrukturen spezifizieren die wichtigsten Daten, die zur Erfüllung der definierten Anforderungen im Rahmen der entwickelten Architektur benötigt werden. Spezifikationen

Das Kapitel beginnt mit einem Abschnitt, in dem die einzelnen Komponenten kurz und übersichtlich vorgestellt werden. Die Übersicht dient lediglich als Rahmen für die detaillierten Spezifikationen, die in Abschnitt 5.2 (Seite 96) folgen. Das Zusammenspiel aller Komponenten ergibt die Gesamtarchitektur des Systems zum integrierten simulationsbasierten Assessment. Um die Funktionsweise der entwickelten Architektur zu verdeutlichen, werden die wichtigsten statischen und dynamischen Aspekte in Abschnitt 5.3 (Seite 148) noch einmal graphisch dargestellt und erläutert. Übersicht

5.1 Komponenten-Übersicht

Ausgangsbasis

Ausgangsbasis für die Abgrenzung der Architektur-Komponenten sind die Ziele, Funktionen und Anforderungen, die in der *Systemdefinition* (Kapitel 4, Seite 69) bestimmt wurden sowie die im Kapitel 3 (Seite 25) vorgestellten Modelle und Standards.

Komponenten

Verantwortlichkeit

Ziel-Zuordnung

Um die Entwicklung der einzelnen Komponenten des SISA besser verfolgen und in das Gesamtsystem einordnen zu können, zeigt Abbildung 5.1 die einzelnen Komponenten in einer Übersicht. Jede Komponente in Abbildung 5.1 besitzt innerhalb des Gesamtsystems eine definierte Verantwortlichkeit und kooperiert zur Erfüllung ihrer Aufgaben mit anderen Komponenten (in dieser Abbildung sind zur Vereinfachung nur die wichtigsten Verbindungen zwischen den Komponenten eingetragen). Eine kurze Erklärung der Verantwortlichkeiten und Kooperationen findet sich in Tabelle 5.1 (Seite 98). Abbildung 5.2 (Seite 99) veranschaulicht die Verbindungen der einzelnen Komponenten mit den definierten Funktionen und Zielen des SISA.

Die Begründungen zu den Abgrenzungen der Komponenten sowie die genauen Spezifikationen der Komponenten, d. h. die Definition der Schnittstellen und Datenstrukturen, finden sich im folgenden Abschnitt.

5.2 Komponenten-Entwicklung

In diesem Abschnitt erfolgt die Spezifikation der einzelnen Komponenten. Die Unterabschnitte spezifizieren jeweils eine der Komponenten und sind ihrerseits in drei Teile gegliedert: im ersten Teil, der Komponenten-Abgrenzung, wird die Verantwortlichkeit der Komponente innerhalb des Gesamtsystems festgelegt; der zweite Teil spezifiziert die Dienste, die von der Komponente zur Erfüllung ihrer Aufgaben anzubieten sind; die Spezifizierung wichtiger Datenstrukturen der Komponente erfolgt im dritten Teil jedes Unterabschnitts.

Da die Verwaltung und Bereitstellung von Metadaten eine zentrale Aufgabe innerhalb der SISA-Architektur besitzt, beginnt die Beschreibung der Komponenten mit dem Katalogmanager.

5.2.1 Katalogmanager

5.2.1.1 Komponenten-Abgrenzung

Metadaten

Eine Funktion des SISA ist die Bereitstellung von Metadaten über vorhandene Datensätze (/F40/). Werden Metadaten nicht nur für vorhandene Daten vorgehalten, sondern auch für alle anderen SISA-Ressourcen[1], so können über die Bereitstellung von Metadaten auch die Funktionen zur Ressourcen-Liste (/F60/)

[1] Vergleiche Abb. 4.2, Seite 76.

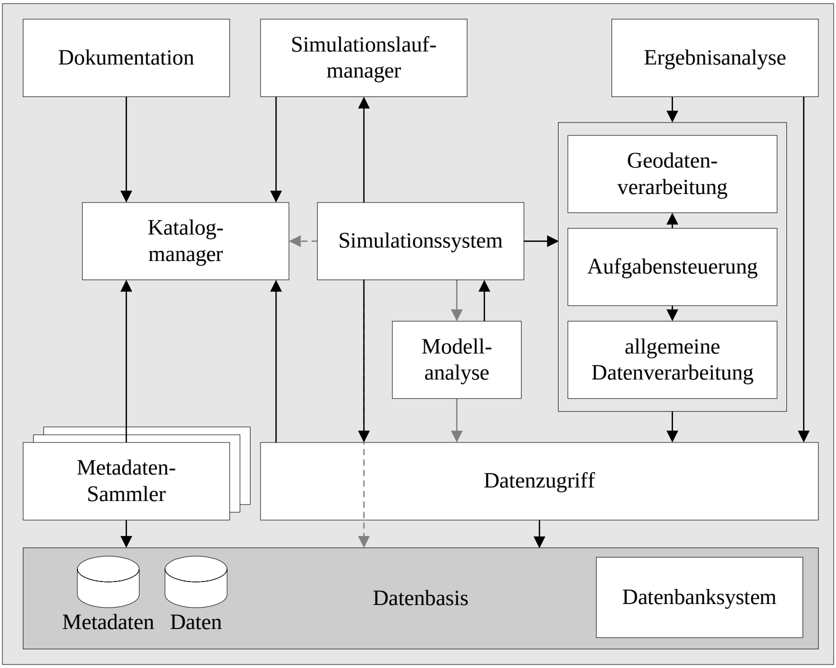


Abbildung 5.1: Übersicht der SISA-Komponenten. Kern des Systems ist das Simulationssystem. Zur Berechnung neuer Ergebnisse verwendet das Simulationssystem Daten aus der Datenbasis, die es über die Datenzugriffskomponente bezieht. Welche Datensätze und Einstellungen für einen Simulationslauf verwendet werden sollen, erfährt das Simulationsmodell vom Simulationslaufmanager. Die für einen Datenzugriff notwendigen Informationen (z. B. Dateinamen) erhält die Datenzugriffskomponente über den Katalogmanager. Metadaten, die nicht über die Benutzungsoberfläche des Katalogmanagers eingegeben wurden, können vom Metadaten-Sammler automatisch in die Komponente übernommen werden. Grundlegende Funktionen zur Geodatenverarbeitung werden von einer gesonderten Komponente angeboten, so dass sowohl das Simulationsmodell als auch der Nutzer des Systems über die Dienste der Aufgabensteuerung auf diese Funktionen zugreifen können. Weitere Datenverarbeitungsfunktionen, die unabhängig von einem konkreten System realisiert und wieder verwendet werden können, werden über die Komponenten der allgemeinen Datenverarbeitung angeboten. Bei einer Modellanalyse (z. B. einer Unsicherheitsanalyse) wendet sich das Simulationssystem nicht direkt an die Datenzugriffskomponente. In diesem Fall bezieht das Simulationssystem alle Daten über die Modellanalyse-Komponente. Die Verbindung der Komponenten mit den funktionalen Anforderungen des SISA können der Übersicht in Abb. 5.2 entnommen werden.

Komponente	Verantwortlichkeit
Katalogmanager	Verwaltung und Bereitstellung von Metadaten über SISA-Ressourcen.
Metadaten-Sammler	Durchsuchung eines Rechners nach Dateien mit Metadaten und automatische Weitergabe der gefundenen Informationen an den Katalogmanager.
Simulationssystem	Berechnung, Speicherung und Weitergabe von Simulationsergebnissen.
Datenzugriff	Lesender und schreibender Zugriff auf Daten und Transformation zwischen verschiedenen Daten-Formaten.
Datenbanksystem	Verwaltete Speicherung von Assessment-Daten.
Simulationslauf-manager	Verwaltung simulationslaufbezogener Einstellungen und Bereitstellung dieser Informationen für die Simulationssystem-Komponente.
Geodatenverarbeitung	Verarbeitung geographischer Daten und die Bereitstellung einer Schnittstelle zu eigenständigen GIS.
Datenverarbeitung	Bereitstellung allgemeiner, wieder verwendbarer Datenverarbeitungsdienste.
Aufgabensteuerung	Programmgesteuerter Aufruf anderer Dienste des SISA (z. B. Dienste zur Datenvorverarbeitung und Datennachbearbeitung).
Ergebnisanalyse	Unterstützung bei der Analyse von Simulationsergebnissen (z. B. bei der Visualisierung und der statistischen Auswertung von Datensätzen).
Modellanalyse	Unterstützung bei der Analyse des Simulationsmodells (insbesondere bei einer Sensitivitäts- oder Unsicherheitsanalyse).
Dokumentation	Dokumentation und Verwaltung wichtiger Assessment-Informationen (z. B. Angaben über Projekte und Hinweise zum Verständnis von Assessments).

Tabelle 5.1: Komponenten und ihre Verantwortlichkeiten.

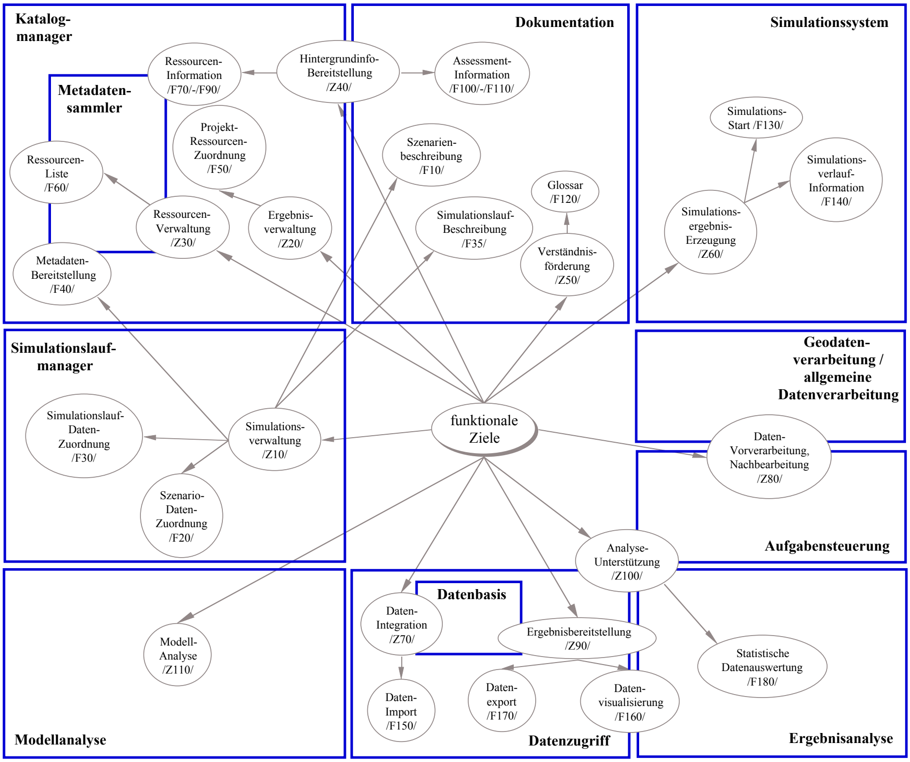


Abbildung 5.2: Übersicht der Verbindung zwischen SISA-Komponenten und SISA-Zielen. Eine kurze Erklärung zentraler Komponenten findet sich in der Beschreibung zu Abbildung 5.1 (Seite 97). Nähere Informationen finden sich im Abschnitt der Komponenten-Entwicklung (5.2, Seite 96).

und zur Bereitstellung grundlegender Informationen zu eingesetzter Software (/F70/), erzeugten Daten (/F80/) und vorhandenen Dokumenten (/F90/) realisiert werden. Die Bereitstellung von Metadaten für alle Ressourcen unterstützt daher auch die Verwaltung von Ressourcen (/Z30/) und die Bereitstellung von Hintergrundinformationen (/Z40/). Die Zuordnung von Projekten zu Ressourcen (/F50/) im Rahmen der Ergebnis-Verwaltung (/Z20/) kann ebenfalls über die Speicherung von Metadaten erreicht werden.

OGC Dienste

Verantwortlichkeit

Die Katalogisierung von Ressourcen kann als weitgehend unabhängig realisierbarer Teil der SISA-Funktionalität betrachtet werden, für den darüber hinaus spezielle Dienste innerhalb der Spezifikationen des OpenGIS-Konsortiums (OGC) definiert sind. Aus diesen Gründen wird ein Katalogmanager als Komponente in die SISA-Architektur aufgenommen, der für die Verwaltung und Bereitstellung von Metadaten über SISA-Ressourcen zuständig ist.

5.2.1.2 Dienst-Spezifikation

OpenGIS Catalog Services

Ausgangspunkt für die Entwicklung der Dienste des Katalogmanagers sind die Spezifikationen des OpenGIS-Konsortiums, genauer: die in *Topic 13* der *Abstract Specifications* definierten *Catalog Services* (Kottmann, 1999c). Diese Katalogdienste sollen die Organisation von und die Suche nach georäumlichen Ressourcen[2] sowie den Zugriff auf diese Ressourcen unterstützen.

Def. Katalog

Der zentrale Begriff des Katalogs wird bei Kottmann (1999c) wie folgt definiert:

> A Catalog is simply a collection of Catalog Entries that is organized to assist in the discovery, access, and retrieval of geospatial resources that are of interest to the user, especially when the existence or whereabouts of the resource are not known to the user.

georäumliche Ressourcen

Welche Ressourcen als georäumliche Ressourcen (geospatial resources) angesehen werden, verdeutlicht Abbildung 5.3. Demnach gehören sowohl Geodienste als auch Geodaten zu den georäumlichen Ressourcen. Unter den Diensten finden sich die Katalogdienste selbst, die Zugriffsdienste auf Daten und alle weiteren Dienste mit georäumlichem Bezug. Die Geodaten sind in Geodata Collections zusammengefasst und beinhalten sowohl die bereits angesprochenen Features[3] als auch Feature Collections. Der Katalog besteht aus einzelnen Katalog-Einträgen (Catalog Entries), die den Inhalt georäumlicher Datensätze beschreiben. Diese Einträge stellen i. d. R. eine Untermenge eines kompletten Metadatensatzes zu einem georäumlichen Datensatz dar und sind so konstruiert, dass Abfragen auf ihnen durchgeführt werden können. Bei den Catalog

[2] Ressourcen mit Raumbezug.

[3] Repräsentationen realer oder imaginärer Dinge oder Phänomene der realen Welt.

Entries handelt es sich also um Metadaten, die zum Auffinden von Ressourcen genutzt werden.

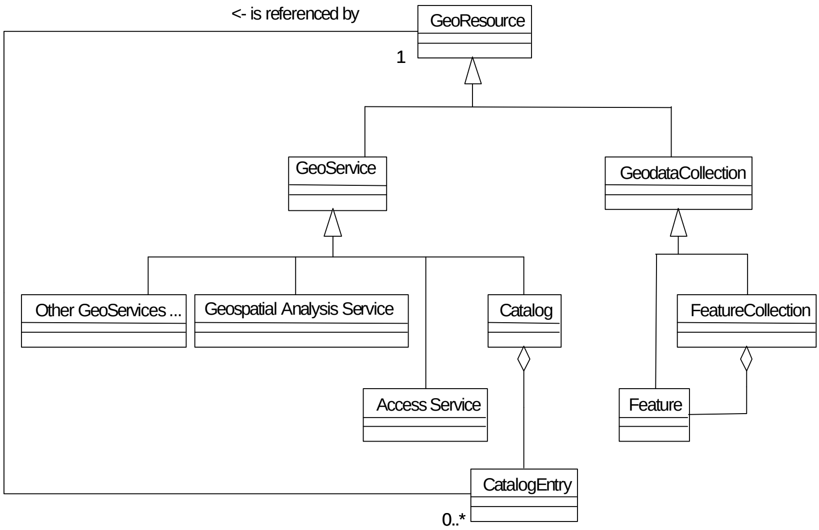


Abbildung 5.3: Georäumliche Ressourcen im Sinne des OpenGIS-Konsortiums. Erklärungen finden sich im Text. Quelle: Kottmann (1999c).

Die Dienste der OpenGIS Catalog Services werden in drei Kategorien eingeteilt (s. auch Abb. 5.4, Seite 102): Dienste

- georesource discovery services
- geodata access services
- other data access services

Die *georesource discovery services* dienen dem Auffinden von georäumlichen Ressourcen und nutzen Metadaten-Repositories, die georäumliche Ressourcen beschreiben und auf diese verweisen. Die *geodata access services* bieten den Zugriff auf georäumliche Daten, die in zugehörigen Daten-Repositories abgelegt sind. Die *georesource disovery services* können auch Datensätze referenzieren, die keine georäumlichen Daten beinhalten; für den Zugriff auf diese Daten sind die Dienste der Kategorie *other data access services* zuständig.

Abbildung 5.4 (Seite 102) zeigt die wichtigsten Klassen, die an der Bereitstellung der Dienste der drei Kategorien beteiligt sind und verdeutlicht die Zusammenhänge zwischen den Klassen. primäre Klassen

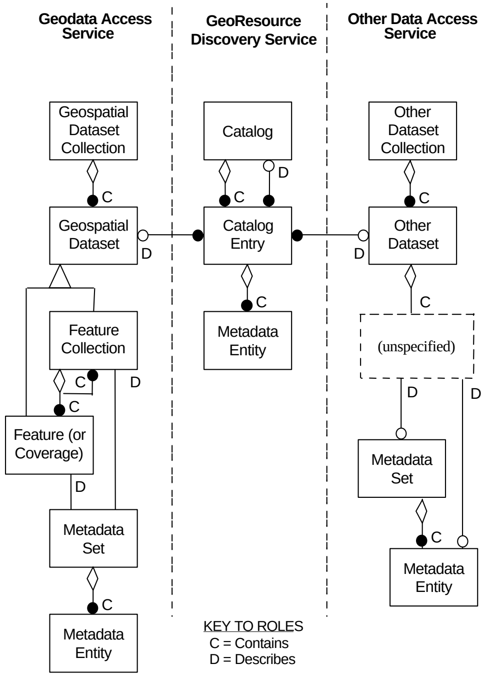


Abbildung 5.4: Primäre, von den *OpenGIS Catalog Services* genutzte Klassen. Erklärungen finden sich im Text. Quelle: Kottmann (1999c).

discovery services

Der Katalog beinhaltet demnach einzelne Katalog-Einträge, wobei jeder Eintrag aus einzelnen Metadaten-Entitäten besteht – einer der Katalog-Einträge kann auch den Katalog selbst beschreiben. Die Metadaten-Entitäten sind Sammlungen so genannter Metadaten-Elemente. Jedes dieser Elemente besteht aus einem Name-Wert-Paar, das eine spezielle Eigenschaft eines Objektes beschreibt – das Paar: Name=„Datum"; Wert=„2003-03-31" ist ein Beispiel für ein solches Metadaten-Element. Die Auswahl konkreter Metadaten-Elemente zur Objektbeschreibung hängt davon ab, welche Metadaten genau gewünscht sind. Aus diesem Grund ist die Klasse als abstrakte Klasse definiert. Die einzelnen Elemente sollten allerdings dem Element-Satz der *OpenGIS Abstract Specifications, Topic 11: Metadata* (Kottmann, 2001) entnommen werden.

Für die Katalog-Klasse und die Klassen der Katalog-Einträge und der Metadaten-Entitäten wird eine Vielzahl mehr oder weniger umfangreicher Funktionen gefordert – angefangen von Funktionen zur Erzeugung und Modifikation der Objekte der einzelnen Klassen bis hin zu genau spezifizierten Abfragen von Katalog-Einträgen. Eine Übersicht der nach Kottmann (1999c) primären *geodata discovery services* findet sich in Tabelle 5.2.

Die Katalog-Einträge können sowohl georäumliche als auch andersartige Datensätze beschreiben. Bei den *georäumlichen Datensätzen* wird davon ausgegangen, dass es sich um OpenGIS Feature Collections oder einzelne Features (oder Coverages) handelt. Die ebenfalls durch das OGC definierten Schnittstellen dieser Klassen (s. Kottmann, 1999e; Kottmann, 2000; Kottmann, 1999b) können also von den *geodata access services* zur Erfüllung ihrer Aufgaben benutzt werden. Die Features und Feature Collections werden über Metadatensätze beschrieben, wobei die Metadaten den Spezifikationen aus Kottmann (2001) entsprechen müssen. Die primären Funktionen der *geodata access services* sind ebenfalls in Tabelle 5.2 aufgeführt. access services

Bei den *other datasets* findet sich eine vergleichbare Strukturierung bei der Assoziation von Metadaten. Eine genaue Spezifikation der Datensätze selbst kann natürlich nicht vorgenommen werden – die entsprechende Klasse bleibt daher unbestimmt. Trotz der nicht näher spezifizierten Klasse gelten für die *other data access services* die gleichen primären Funktionen wie bei den *geodata access services* (vgl. Tab. 5.2).

Geodata Access Service & Other Data Access Service	Geodata Discovery Service
Copy complete dataset	Query catalog service
Retrieve partial dataset	Add catalog entry
Add dataset	Remove catalog entry
Remove dataset	Modify catalog entry
Modify dataset	Copy selected catalog entry
Create iterator through datasets	Create iterator through catalog entries
Query access service	Get catalog entry schema
Get dataset schema	Get service properties
Get service properties	Set service properties
Get service property schema	Get service property schema

Tabelle 5.2: Die primären Funktionen der *OpenGIS Catalog Services*. Eine ausführliche Beschreibung dieser und weiterer Funktionen findet sich in Kottmann (1999c). Spezifikationen zur Implementierung können Nebert (2002) entnommen werden. Quelle der Tabelle: Kottmann (1999c).

Neben den Funktionen, die sich direkt den drei Kategorien (discovery, geodata access, other data access) zuordnen lassen, werden bei Kottmann (1999c) zusätzliche Funktionen aufgeführt, die für alle oder mehrere Dienste wichtig sind. Die Liste enthält u. a. Funktionen zur Bearbeitung von Objekt-Ansammlungen (collections) und zur Behandlung umfangreicher Anfrage-Ergebnisse sowie einen Eintrag, der die Transformation von Daten betrifft. Funktionen zur Daten-Transformation können notwendig werden, wenn die Nutzer gespeicherte Daten-transformation

Daten in einem Format benötigen, das von den *geodata discovery services* oder den *geodata access services* nicht angeboten wird. Als Beispiele für Datentransformationen werden angeführt:

- Konvertierungen des Datenformats (sowohl bezüglich der Werte als auch der Namen von Datenelementen)
- Transformation räumlicher Koordinaten
- semantische Transformation von Datenelementen

Während zur Umsetzung der ersten beiden Punkte bereits Vorschläge der OGC existieren (s. z. B. Kottmann, 1999d), wird die semantische Transformation, also die Umformung der *Bedeutung* von Daten, aufgrund ihrer Komplexität (noch) nicht durch OGC-Spezifikationen beschrieben.

Zur Realisierung der Datentransformation wird der Weg über eine Schnittstellen-Software nach Abbildung 5.5 vorgeschlagen. Die Schnittstellen-Software (oder 'Middleware') ist für alle Datentransformationen zuständig, die für den Datenaustausch zwischen der Anwendungssoftware und den OpenGIS-Diensten notwendig sind. Bei der Erfüllung ihrer Aufgaben nimmt die Schnittstellen-Software Dienste (Data Transformation Services) in Anspruch. Diese Dienste transformieren die Daten in das jeweils benötigte Format: Anwendungsdaten des Formats A werden zur Inanspruchnahme eines OpenGIS-Dienstes über einen Transformationsdienst in ein Format B überführt; die Ergebnisse des OpenGIS-Dienstes werden dann von einem anderen Transformationsdienst vom Ergebnisformat C zum Format D gewandelt, das von der Anwendungssoftware erwartet wird.

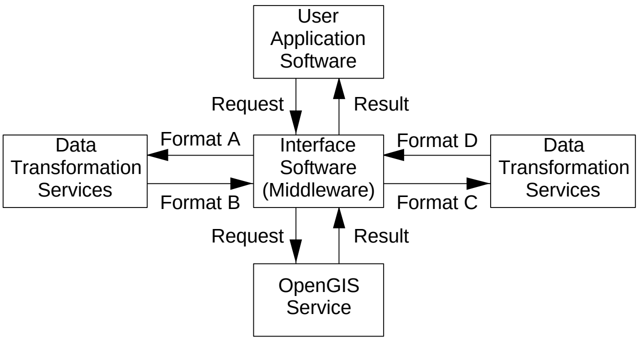


Abbildung 5.5: Nutzung einer Schnittstellen-Software zur Anbindung von *OpenGIS Services* an die Anwendungssoftware. Erklärungen finden sich im Text. Quelle: Kottmann (1999c).

Zur Umsetzung dieses Middleware-Prinzips werden in Kottmann (1999c) verschiedene Möglichkeiten aufgeführt: die Middleware kann, wie in Abbildung

5.5 dargestellt, als eigenständige Software implementiert werden oder Teil bzw. Erweiterung einer der anderen drei Software-Typen (OpenGIS-Dienste, Transformationsdienste, Anwendungssoftware) sein. Es wird allerdings vorgeschlagen, die Middleware als Erweiterung der OpenGIS-Dienste zu sehen und ihr eine Programmierschnittstelle (API) mitzugeben, die sich anlehnt an die APIs, die von den OpenGIS-Diensten angeboten werden.

Um die Geschwindigkeit beim Zugriff auf Daten erhöhen zu können, ist es sinnvoll, die entsprechenden Funktionen von denen des Katalogmanagers zu trennen.[4] Aus diesem Grund wird die Verantwortlichkeit für den Zugriff auf Daten innerhalb der SISA-Architektur einer gesonderten Komponente, der *Datenzugriffskomponente*, übertragen. Weitere Ausführungen zur Datentransformation und zu den *data access services* (vgl. Tab. 5.2, Seite 103) finden sich daher erst im entsprechenden Unterabschnitt (5.2.6, Seite 134). Zugriffs-kompo-nente

Anpassung und Erweiterung der Dienste

Die oben angeführten Prinzipien, Klassen, Dienste und Funktionen bieten einen Rahmen, der beim Entwurf der Katalog-Komponente aus Gründen der Offenheit und Interoperabilität berücksichtigt werden sollte. Eine OGC-konforme Realisierung der Komponente kommt in den meisten Fällen der Entwicklung eines SISA nicht in Frage: die Einhaltung aller Spezifikationen erfordert einen zu hohen Einarbeitungs- und damit Zeit- und Kostenaufwand, der im Rahmen von Modell-Entwicklungen i. d. R. nicht gedeckt werden kann. Aus diesem Grund werden im Folgenden die wichtigsten Dienste und Funktionen spezifiziert, die als Minimal-Ausstattung der Komponente angesehen werden. Hierzu werden zunächst die wichtigsten Katalog-Funktionen identifiziert. Im Anschluss werden die Metadaten-Elemente, die vom Katalog verwaltet werden sollen, spezifiziert. Dienst-Auswahl

Spezifikation der Dienste

Die Schnittstellen-Definition der Katalog-Komponente orientiert sich an der *implementation specifikation* (IS-CAT) der Katalog-Dienste des OGC (Nebert, 2002). In der IS-CAT werden vier Schnittstellen definiert:

Catalog_Service Schnittstelle auf Server-Ebene, die den Zugriff auf die Dienste zur Einrichtung und Verwaltung von Sitzungen (user sessions) bereitstellt

Discovery Operationen, die es dem Nutzer erlauben, Daten, Dienste und andere Ressourcen ausfindig zu machen

[4] Für die Zugriffsdienste kann so beispielsweise eine andere Verteilungsplattform verwendet werden als für die Katalogdienste.

Access Operationen, die den Nutzer beim Zugriff auf die Ressourcen unterstützen, die wiederum über die Operationen des Discovery-Dienstes gefunden werden

Catalog_Manager Operationen zur Verwaltung und Aktualisierung von Katalogen (die Spezifikation der Operationen dieser Schnittstelle ist in Nebert (2002) vorläufig und noch nicht abgeschlossen)

In Anlehnung an die primären Funktionen aus Tabelle 5.2 (Seite 103) und die in Nebert (2002) spezifizierten Operationen werden für die SISA-Architektur die in Tabelle 5.3 zusammengefassten Operationen definiert.

Schnittstelle	Operation	Beschreibung
Catalog Service	initSession	Initialisiert eine Nutzer-Sitzung
	terminateSession	Beendet eine Nutzer-Sitzung
Discovery	query	Abfrage des Kataloges nach bestimmten Kriterien
Access	getAccessInformation	Liefert Informationen zum Zugriff auf eine eindeutig identifizierte Ressource
Catalog Manager	createCatalog	Erzeugt einen neuen Katalog
	createMetadata	Fügt dem Katalog neue Metadaten-Entitäten hinzu
	updateCatalog	Aktualisiert den Inhalt eines gegebenen Kataloges
	deleteCatalog	Löscht den Inhalt des gegebenen Kataloges
	addCatalogEntry	Erzeugt einen neuen Katalogeintrag
	removeCatalogEntry	Entfernt einen Katalogeintrag
	modifyCatalogEntry	Ändert einen Katalogeintrag

Tabelle 5.3: Schnittstellen und Operationen der Katalog-Komponente. Die Schnittstellenbezeichnungen richten sich nach denen von Nebert (2002), die Operationen orientieren sich an Nebert (2002) und Kottmann (1999c) (add-, remove-, modifyCatalog). Weitere Informationen finden sich im Text.

Catalog Service

Die Operationen *initSession* und *terminateSession* der Catalog-Service-Schnittstelle dienen der Koordination verschiedener Nutzer-Sitzungen. Die ISCAT sieht bei der Initialisierung die Rückgabe einer eindeutigen Identifizierungsnummer für die Sitzung vor, die vom Nutzer fortan verwendet werden muss.

Discovery Service

Die zentrale Funktion des *geodata discovery service* ist nach Kottmann (1999c) die Anfrage-Funktion (*query function*). Diese Funktion muss alle Katalog-Einträge innerhalb eines Kataloges finden, die nutzerdefinierten Kriterien entsprechen. Die Syntax und die Semantik von Abfragen wird in den OGC-Spezifikationen zu den Katalog-Diensten (Kottmann, 1999c u. Nebert,

2002) genau festgelegt. Die Berücksichtigung dieser Spezifikationen kann als Maximal-Anforderung an die Funktion angesehen werden. Im Rahmen eines SISA sollte als *Minimal-Anforderung* eine *Anfrage-Funktion* bereitgestellt werden, die einen Katalog-Eintrag aufgrund eines *eindeutigen Ressourcen-Namens* ausfindig macht.

Die Ausgestaltung der Schnittstelle zum Zugriff auf Daten findet hauptsächlich in der Datenzugriffskomponente statt. Der Katalogmanager sollte allerdings diejenigen Informationen zur Verfügung stellen, die für einen Zugriff durch die Datenzugriffskomponente benötigt werden (beispielsweise Informationen über das Datenformat oder das Zugriffsprotokoll). Zu diesem Zweck wird – in Anlehnung an die in Nebert (2002) nicht weiter spezifizierte Operation zum *direct access* – eine Operation mit dem Namen *getAccessInformation* eingeführt. Um eine eindeutige Referenzierung von Ressourcen zu gewährleisten, sollte jede Ressource mit einem eindeutigen Namen, dem so genannten *uniform resource name (URN)* versehen werden (ausführliche Informationen zum Konzept des URN finden sich ab Seite 173 im Kapitel der Realisierung). Für die Erzeugung eines solchen Namens wird ebenfalls eine Operation bereitgestellt. Access

Die Operationen der 'Catalog-Manager'-Schnittstelle dienen der Verwaltung des Katalogs. Die in Tabelle 5.3 aufgeführten Operationen erlauben es, neue Kataloge zu erstellen, vorhandene Kataloge zu aktualisieren und nicht mehr benötigte Kataloge zu löschen. Darüber hinaus stellt die Schnittstelle Operationen zur Erstellung, Entfernung und Änderung von Katalog-Einträgen sowie zum Hinzufügen neuer Metadaten bereit. Catalog Manager

Abbildung 5.6 zeigt die Schnittstellen des Katalogmanagers im Überblick.

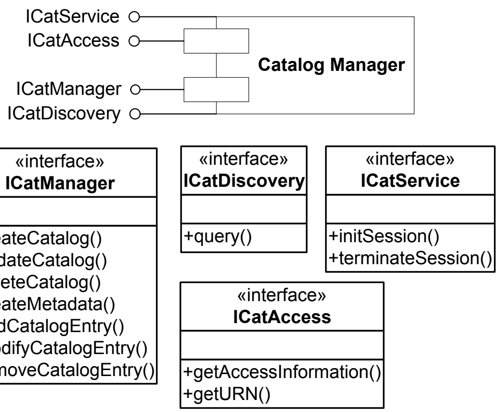


Abbildung 5.6: Schnittstellen und zugehörige Operationen des Katalogmanagers (Catalog Manager).

5.2.1.3 Daten-Spezifikation

Die grundlegende Datenstruktur zur Speicherung des Katalogs besteht aus drei Klassen (vgl. Abb. 5.4, Seite 102); je einer zur Beschreibung der Kataloge, der Katalogeinträge und der Metadateneinträge. Abbildung 5.7 zeigt die für die SISA-Architektur vorgeschlagenen Klassen für den Katalog und die Katalogeinträge. Jeder Katalog erhält einen Namen und besteht aus den einzelnen Katalogeinträgen. Für jede Ressource innerhalb des SISA sollte genau *ein* Katalog-Eintrag existieren, der über den URN identifizierbar ist.

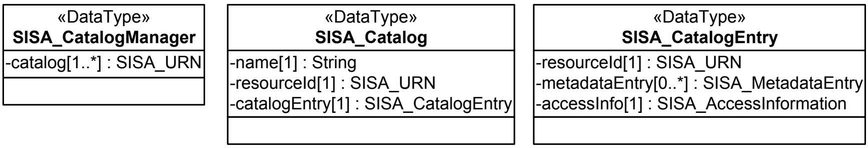


Abbildung 5.7: Datenmodell des Katalogs und der Katalogeinträge.

Für die Spezifikation der Metadateneinträge müssen zunächst die Metadaten-Elemente ausgewählt werden, die für jede SISA-Ressource zu erheben sind.

Spezifikation der Metadaten

Ziel

Ein Katalogeintrag sollte nach Kottmann (1999c) auf einer relativ hohen Abstraktionsebene Antworten auf die folgenden sechs Frageworte geben:

1. Wo – Region der Erde, die der Datensatz abdeckt
2. Was – thematische Schlüsselwörter, Maßstab etc. des Datensatzes
3. Wer – verantwortlicher Ansprechpartner zum Datensatz
4. Wann – Datum der Erzeugung des Datensatzes und evtl. Datum genutzter Ursprungsdaten
5. Wie – Hinweise bezüglich des Zugriffs auf den Datensatz
6. Warum – Informationen zur beabsichtigten Nutzung des Datensatzes

Elemente

Mittels welcher Metadaten-Elemente diese Fragen zu beantworten sind, legt Kottmann (1999c) nicht fest – die Elemente sollten aus Gründen der Interoperabilität und der breiten Anwendbarkeit aber kompatibel zu Metadaten-Standards sein.

Ressourcen

Abgesehen von der Frage nach dem ‘wo’, also den räumlichen Aspekten, sollten die oben genannten Fragen nicht nur für Geodaten, sondern für alle Ressourcen eines SISA (s. Abb. 4.2, Seite 76) beantwortet werden können. Hier stellt sich die Frage nach den Metadaten-Elementen, die zur Beschreibung einer Ressource verwendet werden sollten. Im Folgenden werden wichtige Metadaten-Standards kurz beschrieben.

Metadaten für Geodatensätze –ISO/DIS 19115

Das OGC bezieht sich bei den Metadaten für georäumliche Daten auf die Dokumente der ISO – genauer: auf die Arbeiten des technischen Ausschusses für geographische Informationen der ISO (ISO/TC 211, s. Seite 42). Aus diesem Grund beinhaltet Topic 11 der OpenGIS Abstract Specifications (Kottmann, 2001), das sich mit der Spezifikation von Metadaten beschäftigt, den ISO-Standard ISO/DIS 19115.

Elemente

ISO/DIS 19115 definiert die Metadaten-Elemente in einem *data dictionary*, das insgesamt 409 Einträge enthält. Die Metadaten-Elemente werden zu Entitäten verbunden, welche dann in thematisch getrennten 'Paketen' (Abschnitte im Data Dictionary) verwendet werden. Eine Übersicht der insgesamt 15 definierten Pakete findet sich in Tabelle 5.4. Das Paket 'Constraint Information' beinhaltet beispielsweise Elemente zur Beschreibung rechtlicher, sicherheitsbezogener oder sonstiger Einschränkung bezüglich des Zugriffs oder der Nutzung der Ressource; die Elemente zum Paket 'Data Quality Information' betreffen Informationen über die 'Abstammung' eines Datensatzes, einzelne Verarbeitungsschritte bei der Generierung des Datensatzes und evtl. verwendete Evaluierungsmethoden und deren Ergebnisse. Für die Implementierung der Metadaten stellt die ISO/DIS 19115 insgesamt 27 Tabellen mit Kodierungsinformationen bereit. Darüber hinaus wird innerhalb der Norm auf zahlreiche andere ISO-Standards verwiesen (insbesondere im Paket *units of measure*).

Metadata entity set information	Portrayal catalogue information
Identification information	Distribution information
Constraint information	Metadata extension information
Data quality information	Application schema information
Maintenance information	Extent information
Spatial representation information	Citation and responsible party information
Reference system information	
Content information	Units of measure

Tabelle 5.4: Pakete des Metadaten-Standards ISO/DIS 19115. Weitere Informationen finden sich im Text. Quelle: Kottmann (2001).

Verpflichtung

ISO/DIS 19115 unterscheidet bei den Elementen im Data Dictionary zwischen *zwingend*, *bedingt* und *optional* auszufüllenden Elementen. Nur die zwingend erforderlichen Elemente müssen für jeden Datensatz angegeben werden; bei den bedingten Elementen hängt die Notwendigkeit der Angabe von der Art des Datensatzes, d. h. vom Inhalt eines anderen Elementes ab; die Belegung optionaler Elemente ist hingegen freigestellt.

Kernsatz

ISO/DIS 19115 definiert einen sehr umfangreichen Satz an Metadaten-Elementen, von denen i. d. R. nur eine Untermenge genutzt wird. Zur Dokumen-

tation georäumlicher Datensätze definiert ISO/DIS 19115 aber einen Satz von Metadaten-Elementen, die auf jeden Fall berücksichtigt werden sollen: die so genannten *Kern-Metadaten (core metadata)*. Diese Metadaten dienen nach Kottmann (2001) der Beantwortung der folgenden Fragen:

- Gibt es einen Datensatz zu einem bestimmten Thema ('was')?
- ... für einen bestimmten Ort ('wo')?
- ... für einen bestimmten Zeitpunkt bzw. Zeitraum ('wann')?
- Welchen Kontaktpunkt gibt es, um mehr über den Datensatz zu erfahren bzw. ihn zu bestellen ('wer')?

Eine Übersicht der zu den Kern-Metadaten gehörenden Daten ist in Tabelle 5.5 zu finden. Einige der dort aufgeführten Daten werden direkt durch ein Metadaten-Element repräsentiert: der *dataset title* ist beispielsweise ein Metadaten-Element, dessen Wert mit einem frei wählbaren Text (Freitext) belegt werden kann. Andere Daten müssen unter Berücksichtigung von spezifizierten Kodierungsvorschriften eingetragen werden: *dataset language* ist ein Beispiel für ein solches Datenelement, dessen Wert einer anderen Norm (der ISO 639-2) folgen muss. Wieder andere Daten werden durch mehrere Werte repräsentiert: so kann *geographic location* durch insgesamt vier Werte implementiert werden (westliche und östliche Begrenzung der geographischen Länge sowie nördliche und südliche Begrenzung der geographischen Breite).

Dataset title (M)	Spatial representation type (O)
Dataset reference date (M)	Reference system (O)
Dataset responsible party (O)	Lineage statement (O)
Geographic location of the dataset (C)	Online resource (O)
Dataset language (M)	Metadata file identifier (O)
Dataset character set (C)	Metadata standard name (O)
Dataset topic category (M)	Metadata standard version (O)
Spatial resolution of the dataset (O)	Metadata language (C)
Abstract describing the dataset (M)	Metadata character set (C)
Distribution format (O)	Metadata point of contact (M)
Additional extent information for the dataset (O)	Metadata date stamp (M)

Tabelle 5.5: Kern-Metadaten der ISO/DIS 19115 für georäumliche Datensätze. Die mit einem 'M' (mandatory) gekennzeichneten Daten müssen angegeben werden, unter gewissen Umständen verpflichtende Daten sind mit einem 'C' (conditional) gekennzeichnet und mit einem 'O' (optional) markierte Daten sind optional. Nähere Informationen finden sich im Text. Quelle: Kottmann (2001).

Abbildung 5.8 (Seite 112) zeigt alle Datenelemente der Kern-Metadaten, die für einen Datensatz zwingend ausgefüllt werden müssen. Bereits die Einschränkung auf die zwingenden Elemente zeigt die vielfältigen Verknüpfungen der in ISO/DIS 19115 definierten Klassen, Elemente und Kodierungsspezifikationen: Die Beschreibung des Themas des Datensatzes (Element 'topicCategory') muss beispielsweise unter Verwendung der genormten Kodierungsliste erfolgen, die Sprache muss nach ISO 693-2[5] kodiert werden und die Kodierung des Datensatzes selbst sollte möglichst nach ISO 10646-1[6] erfolgen. Abhängigkeiten

Die vielfältigen Abhängigkeiten und Spezifikationen machen die Metadaten auf der einen Seite sehr gut austauschbar und abfragbar (d. h. sehr interoperabel). Die Metadaten für georäumliche Ressourcen sollten daher, sofern möglich, gemäß ISO/DIS 19115 erfasst werden. Auf der anderen Seite ist die Implementierung und Erhebung der Metadaten recht aufwendig. Die Implementierung des Standards ist auch aufgrund der hohen Anzahl möglicher Datenelemente sehr arbeitsintensiv und kann nur bei entsprechend vorhandenen personellen Ressourcen gewährleistet werden. Aus diesem Grund sollte ein alternativer und weniger komplexer Satz an Metadaten-Elementen an die Seite des ISO-Standards gestellt werden, der mit weniger Aufwand implementiert und erfasst werden kann. Ein solcher Standard ist z. B. der *Dublin Core Metadata Element Set*, der an späterer Stelle (Seite 114) beschrieben wird. Komplexität

Metadaten für Geodatensätze – weitere Normen

Ein weiterer, weit verbreiteter Metadaten-Standard ist der *content standard for digital geospatial metadata (CSDGM)* des *Federal Geographic Data Committee (FGDC)*[7]. Der CSDGM (FGDC, 1998) bietet mit etwa 200 Elementen einen ähnlich umfangreichen Satz an Metadaten-Elementen wie ISO/DIS 19115. Die Angabe einiger Dutzend dieser Elemente ist zwingend vorgeschrieben und etwa 100 gelten als verpflichtend, sofern sie in einem bestimmten Kontext angegebenen werden *können*. Der Element-Satz ist sehr verbreitet, da die Dokumentationen aller durch öffentliche Mittel der USA bezuschussten Daten diesem Standard folgen müssen. Das weit verbreitete Geo-Informationssystem ArcGIS der Firma ESRI unterstützt den CSDGM in seiner Katalog-Software (ArcCatalog) ebenfalls (Vienneau, 2001). FGDC

Da die Entwicklung der ISO/DIS 19115 innerhalb des ISO/TC 211 mit den Arbeiten des FGDC koordiniert ist, bestehen viele Ähnlichkeiten zwischen die-

[5] ISO 693-2: Codes for the representation of names of languages – Part 2: Alpha-3 code.

[6] ISO 10646-1: Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.

[7] Das FGDC ist ein Ausschuss mit Repräsentanten aus 19 US-amerikanischen Behörden (z. B. dem Landwirtschaftsministerium, dem Verteidigungsministerium und der Umweltschutzbehörde), der in Kooperation mit anderen Organisationen für die Entwicklung der nationalen räumlichen Dateninfrastruktur (National Spatial Data Infrastructure, NSDI) zuständig ist.

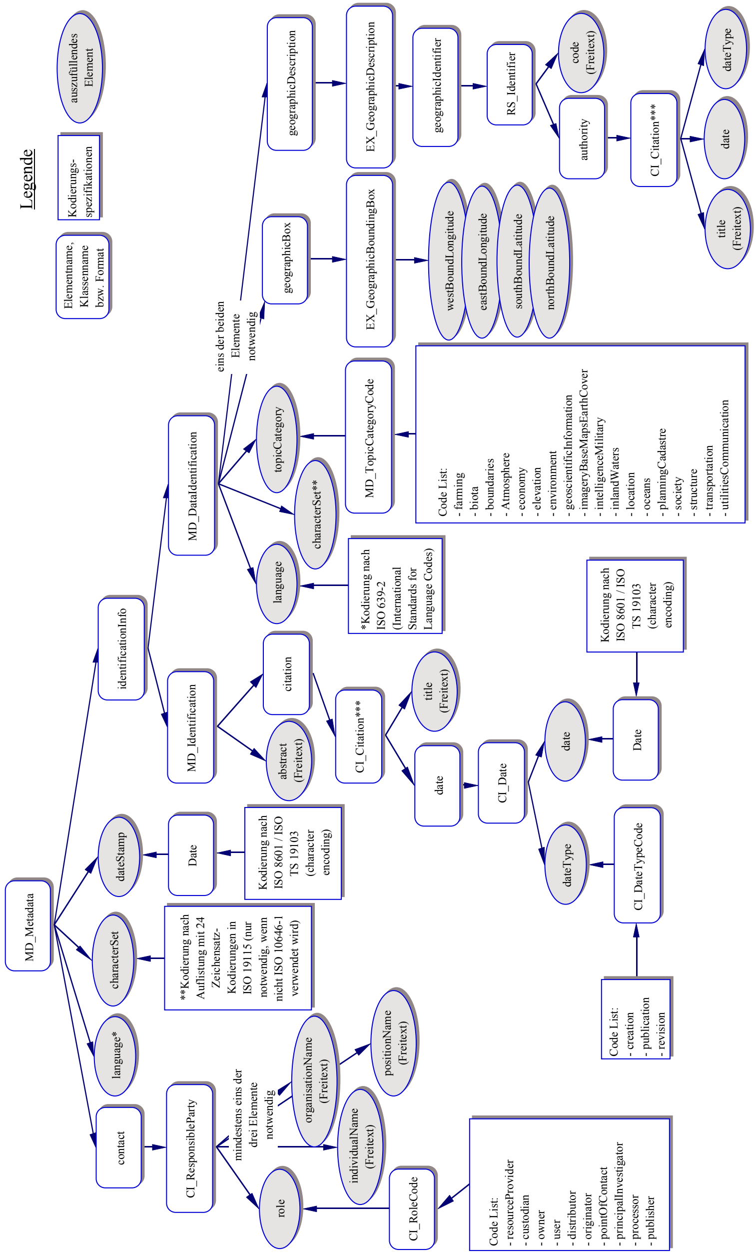


Abbildung 5.8: Minimaler Elementsatz der ISO/DIS 19115 und der mit diesen Elementen in Beziehung stehenden Klassen und Kodierungsspezifikationen.

sen beiden Standards. Aufgrund der Ausrichtung der OGC-Spezifikationen auf die Elemente der ISO/DIS 19115 wird dem ISO-Standard hier allerdings der Vorzug gegeben. Aus den gleichen Gründen wird ISO/DIS 19115 dem ebenfalls etablierten Satz der *ANZLIC-Metadaten-Elementen* (ANZLIC, 2001) vorgezogen, die vom *Australia New Zealand Land Information Council*[8] – dem australischen/neuseeländischen Pendant zum US-amerikanischen FGDC – entwickelt wurden. ANZLIC

Metadaten für Simulationsmodelle

CSCM

Im Rahmen des Projektes *Alexandria Digital Earth Prototype (ADEPT)*[9] wurde ein Ansatz zur Beschreibung berechenbarer Modelle entwickelt: der *Content Standard for Computational Models (CSCM)* (Hill u. a., 2001; ADEPT, 2001).

Ziel

Primäres Ziel des CSCM ist die Bereitstellung von Informationen, die es potentiellen Modellnutzern erlauben, das Modell in einem verteilten, digitalen Katalog zu finden, die Möglichkeit der Anwendung für den eigenen Einsatz zu evaluieren, es zu bekommen, es erfolgreich mit entsprechenden Datensätzen laufen zu lassen und die Ergebnisse zu verstehen (Hill u. a., 2001).

Elemente

Der CSCM, der unter Berücksichtigung der Arbeiten zum ISO/DIS 19115 und des CSDGM erstellt wurde (ADEPT, 2001), enthält rund 160, in zehn 'Abschnitte' gegliederte Elemente. Die Elemente der einzelnen Abschnitte beschreiben ein Modell mit zunehmendem Detaillierungsgrad: angefangen von Elementen zur Identifizierung des Modells (Titel und Version des Modells, verantwortliche Personen etc.) über Informationen zur Verfügbarkeit des Modells bis hin zu Beschreibungen der Modellkalibrierung und -validierung. Tabelle 5.6 (Seite 114) beinhaltet eine kurze Beschreibung der einzelnen Abschnitte.

Verpflichtung

Nicht alle Elemente müssen für jedes Modell angegeben werden: auch bei diesem Standard wird unterschieden zwischen verpflichtend anzugebenden Elementen und solchen, die unter gewissen Bedingungen oder optional auszufüllen sind. Einige Elemente können bzw. müssen auch mehrfach verwendet werden – z. B. Elemente zur Beschreibung einzelner Modellvariablen oder Parameter. Für ein Modell mit einer Eingangsvariablen, die aus einer Datei gelesen werden muss, sind etwa 50 Elementen verpflichtend auszufüllen.

Neben den einzelnen Elementen liefert der CSCM für einige Elemente Kodierungsvorschriften. So gibt es beispielsweise Kodierungslisten zur Beschreibung der Modell-Typologie (Differentialgleichungen, Stochastik, zelluläre Automaten etc.) und zur Klassifizierung des Themas, mit dem sich das Modell beschäftigt (Biologie, Physiologie, Ökologie etc.).

[8]Startseite im Internet: http://www.anzlic.org.au
[9]Startseite im Internet: http://www.alexandria.ucsb.edu

Abschnitt	Beschreibung
Identification Information	Basisinformationen zur Identifizierung des Modells (z. B. Modellname, -version, Identifizierungsnummer, Ansprechpartner).
Indented Use	Beabsichtigte Nutzung des Modells (Thema des Modells) sowie Informationen über das geforderte Wissen (den 'Bildungsstand') zum Verständnis und zur Anwendung des Modells.
Description	Beschreibung des Modells, inkl. des Modellierungsprozesses und der Funktionalität des Modells.
Access & Availability	Informationen zur Verfügbarkeit des Modells (inkl. Zugriffs- und Nutzungseinschränkungen und Angaben zu Bestellung und Kosten).
System Requirements	Anforderungen an das Rechnersystem und den Nutzer zur Durchführung von Modellrechnungen.
Input Data Requirements	Beschreibung der Eingabedaten und ihrer Formate (z. B. Angaben über zeitliche und räumliche Auflösung und Abdeckung der Daten).
Data Processing	Erklärung der auf die Eingabedaten angewendeten Verarbeitungen zur Erzeugung der Ausgabedaten (Angabe zugrunde liegender Formeln und verwendeter Programmiersprachen).
Model Output	Beschreibung der Modellausgaben (Daten bzw. visuelle Ausgaben) (inkl. Angaben über evtl. notwendige Nachbearbeitungen von Ausgabedaten).
Calibration Efforts & Validation	Informationen über Anstrengungen, die zur Modellvalidierung und -kalibrierung unternommen wurden (z. B. Angaben zu Tests, Fallstudien, Validierungsexperimenten).
Metadata Source	Angabe der Person oder Organisation, die für die Generierung der Metadaten verantwortlich ist.

Tabelle 5.6: Übersicht der Abschnitte des *Content Standard for Computational Models*. Quellen: Hill u. a. (2001); ADEPT (2001).

Metadaten für alle Ressourcen – ISO 15836 (Dublin Core)

Info-Res-sourcen

Der *Dublin Core Metadata Element Set* ist ein Metadaten-Standard zur Beschreibung von *Informationsressourcen* unterschiedlichster Bereiche.[10] Ziel bei der Erstellung des Standards waren eine einfache Erzeugung und Wartung der Metadaten, eine allgemein verständliche Semantik der Metadaten-Elemente, ein internationaler Gültigkeitsbereich und die Erweiterbarkeit der Elemente.

Hinter-grund

Die Entwicklungs- und Standardisierungsbemühungen des Elementsatzes gehen zurück auf einen Workshop, der 1995 in Dublin, Ohio (USA) stattfand

[10] Als 'Informationsressource' wird in diesem Standard 'alles' verstanden, 'was eine Identität besitzt'. Dies ist die Definition, die auch im RFC 2396 (Uniform Resource Identifiers (URI): Generic Syntay) der Internet Engineering Task Force (`http://www.ietf.org`) benutzt wird.

und aus dem sich die *Dublin Core Metadata Initiative (DCMI)*[11] bildete. Dieser Workshop brachte Bibliothekare, Wissenschaftler aus dem Bereich digitaler Bibliotheken, Informationsbereitsteller und Experten aus dem Bereich der Auszeichnungssprachen[12] zusammen, um die Standards zur Suche nach Informationsressourcen zu verbessern. Mittlerweile wurde der Elementsatz von mehreren Standardisierungsorganisationen (CEN, W3C, ANSI) übernommen und ist seit Februar 2003 auch im ISO-Standardwerk als ISO 15836 (ISO, 2003) zu finden.

Der *Dublin Core Metadata Element Set (DCMES)* besteht aus 15 Elementen. In ISO (2003) werden die einzelnen Elemente über einen Element-Namen, eine Kennzeichnung (label) und eine Definition spezifiziert und darüber hinaus mit einem kurzen Kommentar versehen – die Element-Namen sollten ja allgemein verständlich sein. Tabelle 5.7 (Seite 116) gibt eine Übersicht der Elemente des DCMES. Elementsatz

Jedes der Elemente aus dem DCMES ist optional und kann für die Beschreibung einer Ressource beliebig oft wiederholt werden. Die Metadaten-Elemente können darüber hinaus in jeder beliebigen Reihenfolge auftreten. Verpflichtung

Für einige Elemente werden in den Kommentaren *Vorschläge* zur Belegung der Werte gemacht: zur Angabe des Verfassers/Urhebers (creator) und des Verlegers/Herausgebers (publisher) wird beispielsweise die Angabe eines Namens empfohlen. Um eine globale Interoperabilität zu fördern, werden darüber hinaus für einige Elemente kontrollierte Vokabulare vorgeschlagen: zur Angabe der Sprache (language) wird beispielsweise die Verwendung des bereits im Rahmen der Beschreibungen zur ISO/DIS 19115 angesprochenen Standards ISO 639 (Codes for the representation of names of languages) empfohlen. Abhängigkeiten

Grundlegende Metadaten-Elemente

Die vorgestellten Elementsätze für Metadaten über georäumliche Daten und Simulationsmodelle bieten eine sehr ausführliche Beschreibungsmöglichkeit dieser Ressourcen. Auf der anderen Seite ist das Ausfüllen der Elemente sehr zeitaufwendig und grenzt bereits an eine Dokumentation dieser Ressourcen an (die nicht das primäre Ziel des Katalogmanagers ist). Für den Katalogmanager wird daher vorgeschlagen, den einfach implementierbaren und ausfüllbaren DCMES als Grundlage für alle SISA-Ressourcen zu verwenden und ISO/DIS 19115 sowie CSCM als optionale Erweiterung zu betrachten. Abbildung 5.9 zeigt die Klassen zur Speicherung der Metadateneinträge.

[11] Startseite im Internet: `http://www.dublincore.org`
[12] XML, SGML etc.

Element-Name	Beschreibung
Title	Ein der Ressource gegebener Name.
Creator	Eine Entität (z. B. Person, Organisation, Dienst), die primär für die Erstellung des Inhalts der Ressource verantwortlich ist.
Subject	Thema des Inhalts der Ressource (z. B. ausgedrückt durch Schlüsselwörter, kodierte Klassifikationen).
Description	Eine Erklärung des Inhalts der Ressource (z. B. eine Zusammenfassung, ein Inhaltsverzeichnis oder eine Referenz auf eine graphische Repräsentation des Inhalts).
Publisher	Eine Entität, die für die Veröffentlichung/Verfügbarkeit der Ressource verantwortlich ist.
Contributor	Eine Entität, die einen Beitrag zum Inhalt der Ressource geleistet hat.
Date	Datum eines Ereignisses (z. B. der Erzeugung oder Veröffentlichung/Verfügbarkeit) im Lebenszyklus der Ressource.
Type	Die Beschaffenheit oder das Genre des Inhalts der Ressource (Beispiele sind: Datensatz, Dienst, Software).
Format	Die physikalische oder digitale Manifestation der Ressource (z. B. Angabe des Medientyps oder einer Formatkennzeichnung).
Identifier	Eine Referenz auf die Ressource, die innerhalb eines gegebenen Kontextes eindeutig ist.
Source	Eine Referenz auf eine Ressource, von der die vorliegenden Ressource abgeleitet ist.
Language	Die Sprache des geistigen Inhalts der Ressource.
Relation	Eine Referenz auf eine Ressource, die mit der vorliegende Ressource in Beziehung steht.
Coverage	Gültigkeitsbereich des Inhalts der Ressource (z. B. die räumliche und die zeitliche Abdeckung).
Rights	Informationen über die Rechte innerhalb oder an der Ressource (z. B. Angaben zu Kopierrechten).

Tabelle 5.7: Die Metadaten-Elemente der ISO 15836 (Information and documentation – The Dublin Core metadata element set) (ISO, 2003).

5.2.2 Metadaten-Sammler

5.2.2.1 Komponenten-Abgrenzung

zentral vs. dezentral

Die zentrale Verwaltung der Metadaten über das Metadaten-Repository innerhalb des Katalogmanagers bringt Vorteile bei der Suche nach Ressourcen. Andererseits ist es oft von Vorteil, wenn die Beschreibung von Daten an *dem* Ort gespeichert ist, wo auch die Daten selbst gespeichert sind (z. B. in Form einer zusätzlichen Datei). Die Informationen zu einem Datensatz sind dann direkt dort, wo der Datensatz beim ‘Durchsuchen’ von Verzeichnissen gefunden wird

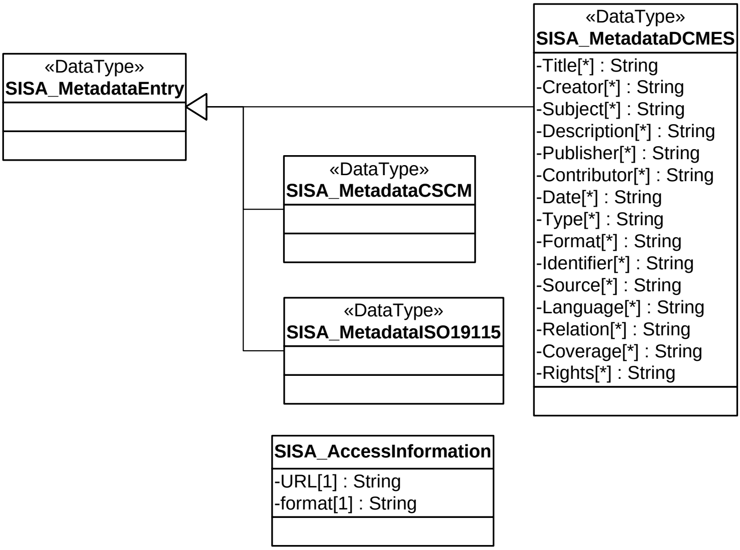


Abbildung 5.9: Datenmodell zur Speicherung von Metadaten. Als Grundlage sollen die Elemente des *Dublin Core Metadata Element Set* (ISO, 2003) verwendet werden. Die Erweiterung um die Sätze der *ISO/DIS 19115* (ISO, 2000) und des *CSCM* (ADEPT, 2001) sollten bei Bedarf als Unterklasse von SISA_MetadataEntry einfügt werden (deren Elementsätze werden aufgrund ihrer Komplexität nicht im Diagramm dargestellt).

und sofort einsehbar[13]; werden die Daten kopiert, in ein anderes Verzeichnis (oder auf einen anderen Rechner) verschoben oder anderen Personen überreicht, können die Metadaten sehr einfach mit kopiert werden. Einige Anwendungen (z. B. Programme zur Textverarbeitung oder Tabellenkalkulation, aber auch GIS) bieten auch die Möglichkeit, Metadaten direkt als Teil des Arbeitsergebnisses (z. B. ein Text-Dokument) zu integrieren. Die dezentrale Speicherung von Metadaten besitzt also auch einige Vorteile.

Harvester

Damit der Nutzer des Systems dezentral angelegte Metadaten nicht zusätzlich in den Katalogmanager eintragen muss, wird die Verwendung von Metadaten-Sammlern vorgeschlagen. Ein Metadaten-Sammler (engl. Metadata Harvester) ist ein Programm, das die Verzeichnisse eines Rechners nach Metadaten durchsucht und diese sammelt ('erntet'). Die zusammengetragenen Metadaten sind darüber hinaus einer zentralen Stelle – in Falle des SISA ist dies der Katalogmanager – mitzuteilen. Eine solcher Sammler sollte auf allen

[13] Sofern ein einfaches Datenformat verwendet wird bzw. die Software zum Lesen komplizierterer Formate bereitsteht.

Rechnern installiert sein, die an der Erzeugung von Daten für ein Projekt beteiligt sind. Abbildung 5.10 verdeutlicht diese Vorgehensweise in graphischer Form.

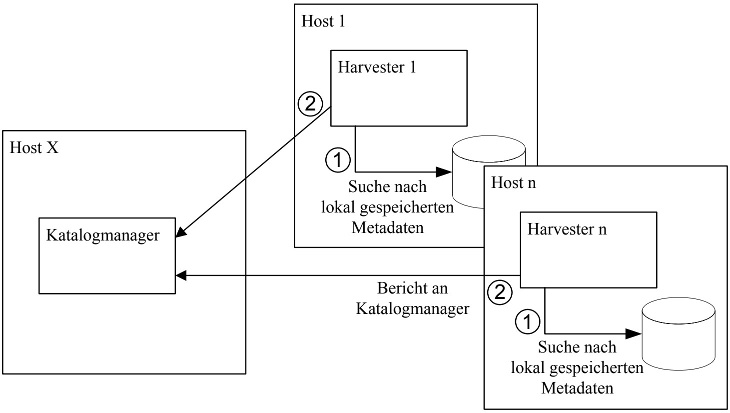


Abbildung 5.10: Prinzip des Metadaten-Sammlers. Dezentral an einem Host erfasste und gespeicherte Metadaten werden vom Metadaten-Sammler (Harvester) gesucht und zusammengetragen (Schritt 1). Im Anschluss an die Suche, die regelmäßig oder auf Anforderung stattfinden kann, werden die Daten dem Katalogmanager zur Verfügung gestellt (Schritt 2). Auf jedem Host, der an einem Projekt beteiligt ist (Host 1 bis Host n), sollte für diese Zwecke ein Harvester installiert sein.

Verantwortlichkeit

Ein mit dem Katalogmanager zusammenarbeitender Metadaten-Sammler ist damit verantwortlich für die Durchsuchung eines Rechners nach Dateien mit Metadaten und die automatische Weitergabe der gefundenen Informationen an den Katalogmanager.

5.2.2.2 Dienst-Spezifikation

Die von einem Metadaten-Sammler anzubietenden Operationen ergeben sich direkt aus den beiden in Abb. 5.10 erklärten Schritten: der *Suche und Zusammentragung* von Metadaten und dem *Bericht* der gefundenen Informationen an den Katalogmanager. Über diese beiden Funktionen des Sammlers hinaus, kann es sinnvoll sein, die auf einem Host – ebenfalls dezentral – vorhandenen Metadaten in einem lokalen Verzeichnis (Repository) vorzuhalten. Ein solches Repository erleichtert den Überblick über die auf einem Host vorhandenen Daten, ohne die Notwendigkeit mit dem Katalogmanager in Verbindung zu treten. Entsprechende Operationen zur Einrichtung und Abfrage eines solchen Verzeichnisses

sind daher ebenfalls von der Komponente anzubieten. Abbildung 5.11 zeigt die Schnittstellenspezifikation des Metadaten-Sammlers als UML-Diagramm.

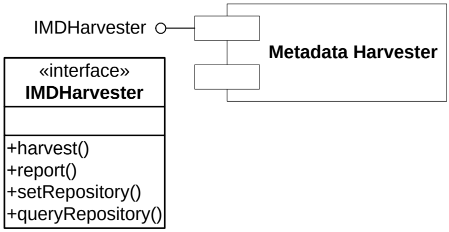


Abbildung 5.11: Schnittstelle des Metadaten-Sammlers. Die Komponente besitzt lediglich eine Schnittstelle (IMDHarvester), die sowohl die Operation zum Einsammeln von Metadaten (harvest) als auch die Operation zur Versendung der gesammelten Metadaten an den Katalogmanager (report) bereitstellt. Falls die Speicherung der gesammelten Daten in einem lokalen Verzeichnis (Repository) gewünscht ist, kann ein solches über zusätzliche Operationen gesetzt (setRepository) und abgefragt (queryRepository) werden.

5.2.2.3 Daten-Spezifikation

Das Datenmodell für die Sammlung und den Bericht der Metadaten ist abhängig vom lokal verwendeten Metadaten-Elementsatz. Um eine nahtlose Integration der Metadaten in den Katalog des Katalogmanagers zu ermöglichen, sollten sich die Metadaten-Elemente an denen orientieren, die im Katalogmanager verwendet werden (s. Unterabschnitt 5.2.1, Seite 108). Metadaten

Da der Metadaten-Harvester mit dem Katalogmanager kooperiert, benötigt er einen Verweis auf diese Komponente. Zum Zugriff auf das evtl. vorhandene Repository ist ebenfalls ein entsprechender Verweis (z. B. auf den Namen einer Datei oder einen verantwortlichen Dienst) notwendig. Abbildung 5.12 zeigt die resultierende Klasse in UML-Notation. Attribute

5.2.3 Dokumentation

5.2.3.1 Verantwortlichkeit

Zu den Zielen eines SISA gehört es, Simulationsläufe zu verwalten (/Z10/), Hintergrund-Informationen zu Assessments bereitzustellen (/Z40/) sowie Hilfestellungen zum Verständnis des Problembereichs zu geben (/Z50/). Die Verwaltung der in diesem Zusammenhang notwendigen Informationen über verwendete Software und verwendete bzw. erzeugte Daten und Dokumente (/F70/-/F90/) wird bereits vom Katalogmanager übernommen (s. Unterabschnitt Abgrenzung

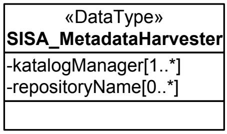


Abbildung 5.12: Datenmodell des Metadaten-Sammlers. Neben einem – implementierungsabhängigen – Verweis auf den Katalogmanager muss der Metadaten-Sammler noch den Namen eines optional anzulegenden Verzeichnisses (Repository) speichern können. Die Datentypen der Attribute sind abhängig von der konkreten Realisierung und werden daher an dieser Stelle nicht spezifiziert.

5.2.1, Seite 96). Während diese Informationen in Form von Metadaten verwaltet werden, ist für die Bereitstellung von Informationen über durchgeführte Simulationsläufe (/F35/), zugrunde liegenden Szenarien (/F10/), beteiligte Personen (/F100/) und durchgeführte Projekte und Studien (/F110/) sowie das geforderte Glossar (/F120/) eine direkte Datenhaltung notwendig (s. 'Hintergrunddaten' in Abb. 4.4, Seite 85). Die Verantwortlichkeit für diese Funktionen und Daten wird einer separaten Komponente übertragen: der *Dokumentationskomponente*. Über die genannten Informationen hinaus sollte die Dokumentationskomponente Modellbetreibern Hintergrundinformationen über die Bedienung des Systems bereitstellen. Aus dieser Abgrenzung heraus ergibt sich die nachstehende Verantwortlichkeit der Komponente. Die Dokumentationskomponente ist verantwortlich für die Bereitstellung und Verwaltung grundlegender Informationen über durchgeführte bzw. in Bearbeitung befindliche Assessments, den Problembereich des modellierten System sowie die Bedienung des Systems.

Verantwortlichkeit

5.2.3.2 Dienst-Spezifikation

Einige Dienste, die die Dokumentationskomponente zur Verfügung zu stellen hat, ergeben sich direkt aus den ihr zugeordneten funktionalen Anforderungen (vgl. Abb. 5.2, Seite 99). Neben der Bereitstellung eines Glossars (/F120/) sind Daten vorzuhalten für:

Klassen

- beteiligte Personen/Organisationen (/F100/)
- durchgeführte Projekte/Studien (/F110/)
- durchgeführte Simulationsläufe (/F35/)
- verwendete Szenarien (/F10/)

Diese Daten müssen über die Komponente aufgenommen, abgefragt, modifiziert und gelöscht werden können.

Die einzelnen Datensätze können als Einträge in entsprechenden Katalogen betrachtet werden. Aus diesem Grund orientierten sich die Schnittstellendefinitionen der Dokumentationskomponente (s. Abb. 5.13) an den definierten Schnittstellen des Katalogmanagers (Unterabschnitt 5.2.1, Seite 96).

Neben Katalogen für Personen, Organisationen, Projekte, Studien, Szenarien, Simulationsläufe und das Glossar, werden zwei zusätzliche Kataloge vorgeschlagen: einer zur Dokumentation von Arbeitsschritten und einer zur Aufnahme kurzer Anmerkungstexte. Weitere Informationen zu den Katalogen finden sich in den Absätzen der folgenden Daten-Spezifikation.

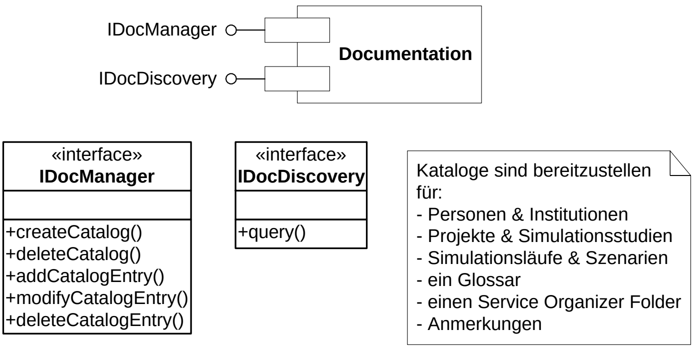


Abbildung 5.13: Schnittstellen der Dokumentationskomponente. Die Namen der Operationen orientieren sich an den Spezifikationen des OpenGIS-Konsortiums (vgl. Schnittstellen-Spezifikation des Katalogmanagers, Abb. 5.6, Seite 107). Erklärungen zu den Informationen, die für die einzelnen Kataloge zu speichern sind, finden sich in den Absätzen zur Daten-Spezifikation.

5.2.3.3 Daten-Spezifikation

Zur Erfüllung der SISA-Anforderungen müssen einige Daten innerhalb des Systems verwaltet und vorgehalten werden. Einige der Klassen, für die Informationen gespeichert werden müssen, wurden bereits bei der Systemdefinition in Unterabschnitt 4.2.5 (Seite 84) identifiziert und in Abb. 4.4 (Seite 85) dargestellt. Demnach sind Attribute zu den folgenden Klassen zu spezifizieren: Übersicht

- Person
- Organisation
- Projekt
- Simulationsstudie
- Simulationslauf
- Szenario

Zur Steigerung der Nachvollziehbarkeit von Assessments werden über diese Klassen hinaus zwei weitere Klassen definiert. Die erste Klasse orientiert sich am so genannten ‘Service Organizer Folder’ (Percivall, 2002) und erlaubt die Dokumentation von Arbeitsschritten, die zur Lösung einer bestimmten Aufgabe notwendig sind. Die zweite Klasse dient zur Speicherung einfacher Notizen, die verwendet werden können, um ein Assessment oder einzelne Ressourcen mit Anmerkungen zu versehen. Die entsprechenden Klassennamen sind:

- Arbeitsschritt
- Anmerkung

Die folgenden Absätze spezifizieren die Attribute der aufgeführten Klassen.

Personen und Organisationen

ISO/DIS 19115

Die über Personen und Organisationen zu speichernden Daten (/F100/) sollten nach den Richtlinien der ISO/DIS 19115 (Kottmann, 2001) kodiert werden, um eine nahtlose Integration von Projektdaten und Metadaten zu ermöglichen. Der Standard schlägt zur Verwaltung von Personen und Organisationen eine Klasse mit dem Namen ‘responsible party’ (verantwortliche Instanz) vor, die Angaben zum Namen und zu Kontaktinformationen vorsieht. Die genaue Klassendefinition wird im Anhang B.1 (Seite 234) in UML-Notation wiedergegeben.

Rollen

Zur Beschreibung der Rolle, die einer Person/Organisation im Rahmen der Metadaten-Erfassung zugeschrieben werden kann, bietet ISO/DIS 19115 eine Kodierungsliste (s. Abb. B.1, Seite 234). Da sich diese Liste auf mögliche Rollen im Zusammenhang mit Metadaten beschränkt, muss sie an dieser Stelle erweitert werden.

Die zentralen Rollen von Personen im Rahmen des simulationsbasierten Assessments können direkt dem OOA-Modell des SISA (Abb. 4.1, Seite 73) entnommen werden: Modellentwickler, -implementierer, -betreiber, Entscheidungsträger, Ressourcenlieferant und Interessent. Als weitere Rolle kommt noch der Kapitalgeber hinzu. Abbildung 5.14 zeigt das resultierende Datenmodell im Überblick.[14]

Projekte und Simulationsstudien

Projekte

Innerhalb des SISA sollten die wichtigsten, auf das Assessment bezogenen Informationen zu Projekten gespeichert werden (/F110/). In ISO 9000 (DIN, 2000) wird der Begriff des Projektes definiert als: „Einmaliger Prozess, der aus einem Satz von abgestimmten und gelenkten Tätigkeiten mit Anfangs- und

[14] ISO/DIS 19115 verwendet den Namen ‘responsible party’ nicht nur für die verantwortlichen Instanzen, sondern auch für die Nutzer oder Eigentümer von Ressourcen (s. CI_RoleCode in Abb. B.1, Seite 234). In Anlehnung an ISO/DIS 19115 – und mit der gleichen Anmerkung versehen – wird auch für den SISA-Datentyp der Name ‘responsible party’ gewählt.

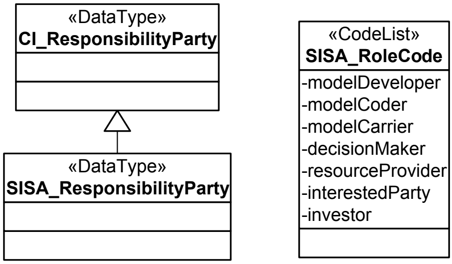


Abbildung 5.14: Datenmodell zur Speicherung von Informationen über Personen und Organisationen. Die Klasse zur Speicherung von Informationen zu Personen und Organisationen (SISA_ResponsibleParty) leitet sich direkt von der entsprechenden Klasse der ISO/DIS 19119 (CI_ResponsibleParty) ab. Die Kodierungsliste zur Angabe der Rolle, die eine Person oder Organisation einnimmt, wird für das SISA entsprechend erweitert (SISA_RoleCode). Die diesbezüglichen Definitionen der ISO sind im Anhang zu finden (Abb. B.1, Seite 234).

Endtermin besteht und durchgeführt wird, um ein Ziel zu erreichen, das spezifische Anforderungen erfüllt, wobei Zeit-, Kosten- und Ressourcenbeschränkungen eingeschlossen sind.“

Das Datenmodell des Umweltdatenkatalogs[15] (Swoboda u. a., 1998; Swoboda u. a., 2000) beschreibt Projekte über die Datenelemente *Projektleiter*, *Beteiligte* und *Erläuterung*.[16] Über diese Angaben hinaus sollte das SISA, in Anlehnung an die ISO-Definition, auf jeden Fall Informationen über den *Anfangs- und Endtermin* sowie das *Ziel* des Projektes bereitstellen. Die Angabe eines *Projekt-Titels* sowie der Verweis auf *weitere Informationsquellen* – Dokumente mit detaillierten Angaben zu Kosten, Ressourcen usw. – sollten ebenfalls über das SISA bereitgestellt werden. UDK Erweiterung

Um eine getrennte Verwaltung von Personen und Organisationen zu erleichtern, sollte zur Aufnahme der Daten jeweils ein separater Katalog erzeugt und verwendet werden.

Für Projekte werden Simulationsstudien mit Simulationsläufen und Ergebnis-Analysen durchgeführt (vgl. Abb. 4.1, Seite 73). Studien sind „wissenschaftliche Untersuchungen über eine Einzelfrage“ (Duden, 1996). In diesem Sinne können Simulationsstudien als ‘kleine’ Projekte (Unterprojekte) betrachtet werden (Ziele von Simulationsstudien können neben der Erstellung eines Assess- Studien

[15] Der Umweltdatenkatalog (UDK) ist ein Metadaten-Informationssystem zum Auffinden umweltrelevanter Datenquellen, die in den öffentlichen Verwaltungen vorhanden sind. Startseite im Internet: http://www.umweltdatenkatalog.de

[16] Im Umweltdatenkatalog werden die gleichen Elemente auch für ‘Vorhaben’ und ‘Programme’ benutzt.

ments z. B. die Durchführung einer Modellvalidierung oder Sensitivitätsanalyse sein). Aus dieser Überlegung heraus entspricht die Datenstruktur zur Beschreibung von Simulationsstudien – erweitert um die *Zuordnung zu Projekten* – der Datenstruktur für die Projekte. Abbildung 5.15 zeigt die UML-Diagramme beider Klassen.

Projekte und Studien sollten, genauso wie Personen und Organisationen, in jeweils getrennten Katalogen verwaltet werden.

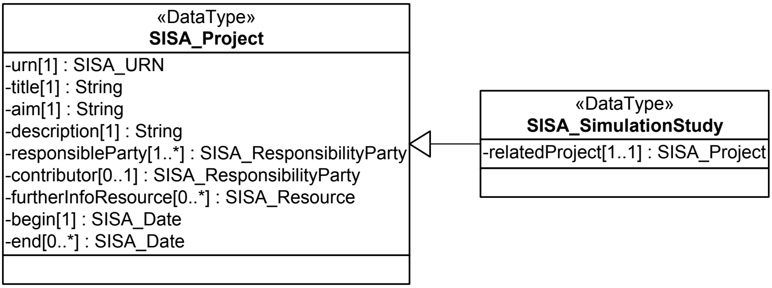


Abbildung 5.15: Datenmodell zur Speicherung von Informationen über Projekte und Simulationsstudien.

Szenarien

Zur Speicherung von Informationen zu Szenarien stellt sich die Frage, was ein Szenario genau ist bzw. ausmacht. Szenarien wurden, in einer auf die Modellierung gerichteten Sichtweise, in Abschnitt 2.2 (Seite 14) definiert als die „in sich konsistenten und plausiblen Annahmen über die zukünftige Entwicklung systembeeinflussender exogener Größen“ (Bossel, 1994). Eine weniger technisch ausgerichtete Definition gibt das IPCC in einem Bericht über Emissions-Szenarien: „Scenarios are images of the future, or alternative futures. They are neither predictions nor forecasts. Rather, each scenario is one alternative image of how the future might unfold.“ (Nakicenovic u. a., 2000)

Dokumente

Die Beschreibung solcher Szenarien erfolgt aufgrund ihrer Komplexität i. d. R. in Form von Berichten oder anderen Publikationen (s. z. B. Nakicenovic u. a., 2000). Innerhalb des SISA sollen als Hintergrundinformation dennoch die wichtigsten Merkmale der verwendeten Szenarien vorgehalten werden.

5 Elemente

Zur Frage welche Hintergrundinformationen für die ‘möglichen Zukünfte’ im SISA zu speichern sind, können die ‘fünf prinzipiellen Elemente’ herangezogen werden, aus denen nach Alcamo (2001) ein typisches Szenario im Rahmen von Umweltstudien besteht:

Beschreibung schrittweiser Änderungen Beschreibung des sich schrittweise ändernden, zukünftigen Status von Gesellschaft und Umwelt (Beschreibung über Indikatoren)

exogene Einflussfaktoren (driving forces) Schlüsselfaktoren bzw. Determinanten, die den Gang der schrittweisen Änderungen hauptsächlich beeinflussen

Basisjahr Markierung des Beginns des Szenarios (in quantitativen Szenarien oft das aktuellste Jahr, für das Daten vorhanden sind)

Zeithorizont und Schrittweite Markierung des am weitesten in der Zukunft liegenden Jahres, das vom Szenario abgedeckt wird (Zeithorizont) sowie das Zeitintervall zwischen zwei Beschreibungsschritten

Entwicklungsgeschichte (storyline) Erzählende Beschreibung des Szenarios, die die zentralen Punkte und Trends des Szenarios sowie deren Beziehung zu den exogenen Einflussfaktoren enthält

Im SISA sollten demnach Angaben zu *Basisjahr*, *Zeithorizont* und *Schrittweite* vorhanden sein sowie zusammenfassende Beschreibungen der *exogenen Einflussfaktoren*, der *Entwicklungsgeschichte* und der sich schrittweise vollziehenden *Änderungen*. Ein Verweis auf *weiterführende Informationsquellen* sollte ebenfalls möglich sein.

Um die Ergebnisse von Simulationsstudien richtig einschätzen und bewerten zu können, sollte das SISA, neben den Angaben zu den fünf prinzipiellen Elementen, weitere Informationen über die zugrunde liegenden Szenarien bereitstellen. Erweiterung

Für Fink (2002) ist die Entwicklung von Szenarien u. a. mit zwei zentralen Fragen verbunden: Was soll mit Hilfe der Szenarien gestaltet werden? Was soll durch die erstellten Szenarien erklärt werden? Zumindest eine kurze, zusammenfassende Antwort auf diese Fragen nach dem *Ziel eines Szenarios* sollte das SISA speichern können. Eine weitere wichtige Frage, die nach Fink (2002) vor dem Beginn der Szenarienentwicklung beantwortet werden sollte, ist die nach dem *räumlichen Fokus* eines Szenarios; auch diese Information sollte im SISA vorhanden sein. Abbildung 5.16 zeigt das resultierende Datenmodell zur Speicherung von Szenarien-Informationen.

Simulationsläufe

Zu einer Simulationsstudie gehören i. d. R. mehrere Simulationsläufe. Jeder dieser Läufe wird mit einem bestimmten Ziel und unter Verwendung genau festgelegter Randbedingungen durchgeführt. Zur Sicherstellung der Nachvollziehbarkeit und Reproduzierbarkeit von Simulationsergebnissen sollten alle Simulationsläufe dokumentiert sein. Zu dieser Dokumentation gehören zumindest Angaben zum *Ziel* eines Simulationslaufes, zum *Verantwortlichen* (d. h. dem Modell-

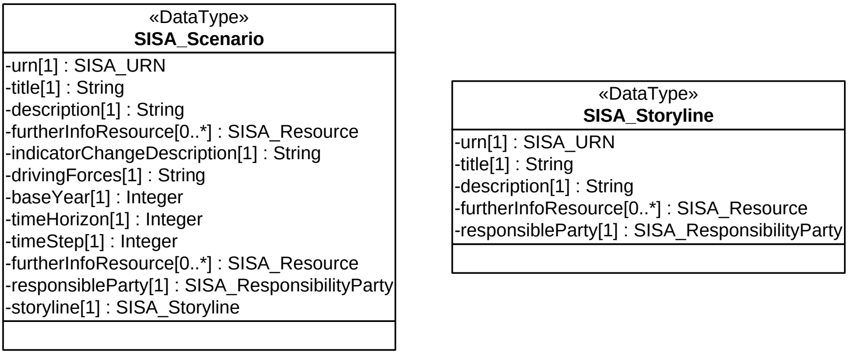


Abbildung 5.16: Datenmodell zur Speicherung von Szenario-Informationen. Da die Erzeugung einer neuen Storyline sehr zeit- und arbeitsaufwendig ist, schlägt Alcamo (2001) die Wiederverwendung bereits vorhandener und akzeptierter Storylines vor. Aus diesem Grund wird die 'Storyline' als eigenständiger Datentyp definiert.

betreiber) und zu *Randbedingungen*, also den zugehörigen Simulationsmodell-Daten, die evtl. über ein zugeordnetes *Szenario* definiert sein können.

Die Zuordnung von Daten zu Simulationsläufen und Szenarien liegt in der Verantwortung des Simulationslaufmanagers, so dass innerhalb der Dokumentationskomponente nur die in Abb. 5.17 dargestellten Informationen vorzuhalten sind.

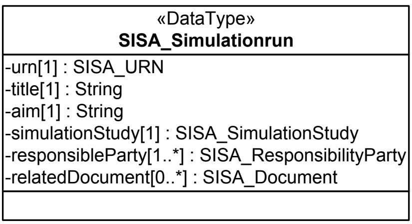


Abbildung 5.17: Datenmodell zur Speicherung von Simulationsläufen.

Service Organizer Folder

Dienst-Referenz

Das OpenGIS-Konsortium führt in der Definition der *Service Architecture* (ISO /DIS 19119, Percivall, 2002), neben der Dienste-Taxonomie und den zugeordneten Beispieldiensten, den so genannten *service organizer folder (SOF)* ein.

Ein SOF ist eine Datenstruktur, die Referenzen auf Diensten beinhaltet, die in bestimmten Situationen sinnvoll eingesetzt werden können. Über diese Struktur können von den Nutzern eines Systems selbst Dienste gruppiert werden, die sie zur Erfüllung einer bestimmten Aufgabe benötigen. Diese Gruppierung kann dann von anderen Nutzern bei der Bearbeitung vergleichbarer Aufgaben eingesetzt werden. Ein SOF stellt also eine Art Dokumentation vorhandener Lösungsmöglichkeiten für bestimmte Problembereiche dar.

Ressourcen

Bei der Durchführung eines Assessments sind vielfältige Aufgaben zu bearbeiten: Modellteile müssen angepasst, Daten in das richtige Format gebracht und Informationen aufbereitet werden. Die hierzu notwendigen Funktionen (Dienste) werden oft über speziell für diesen Zweck geschriebene Werkzeuge bereitgestellt. Da die oben beschriebenen Aufgaben oft nicht automatisiert sind – und sich teilweise auch nicht mit einem vertretbaren Aufwand automatisieren lassen – sollten alle in einer bestimmten Situation anwendbaren Werkzeuge und notwendigen Schritte dokumentiert sein. Eine solche Dokumentation kann erleichtert werden, indem die ursprüngliche *Beschränkung* des SOF auf *Dienste* aufgehoben und die Angabe *aller Ressourcen*, die bei der Lösung einer Aufgabe herangezogen werden können, erlaubt wird.[17]

Aufgaben

Ein SOF, der alle SISA-Ressourcen aufnehmen kann, unterstützt sowohl die Daten-Vorverarbeitung (/Z80/) als auch die Ergebnis-Analyse (/Z100/) und fördert darüber hinaus die Nachvollziehbarkeit von Assessment-Ergebnissen. Abbildung 5.18 zeigt die innerhalb des SOF für die Aufgaben-Dokumentation zu speichernden Attribute. Alle Angaben sollten innerhalb eines eigenen Katalogs, dem SOF-Katalog, verwaltet werden.

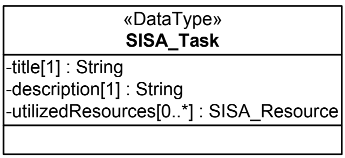


Abbildung 5.18: Datenmodell zur Speicherung von Informationen über die Bearbeitung von Aufgaben.

Anmerkungen

Neben der Nutzung des SOF kann es für Modellbetreiber und Modellentwickler sehr hilfreich sein, kurze Anmerkungen zum SISA direkt innerhalb des System

[17] Die grundsätzliche Möglichkeit die innerhalb des SOF aufgeführten Dienste automatisch abarbeiten zu lassen wird durch die Erweiterung nicht eingeschränkt, sofern die einzelnen Einträge als ‘Dienst’ bzw. ‘Nicht-Dienst’ gekennzeichnet sind.

abzulegen, z. B. Anmerkungen zur Verbesserung des SISA oder zu aufgetretenen Problemen. Um diese Möglichkeit zu bieten wird ein Datentyp für Anmerkungen vorgeschlagen, der als Eintrag in einem weiteren Katalog (dem Anmerkungskatalog) verwendet werden kann. Um die Einträge des Katalogs einfacher auflisten zu können, muss jede Anmerkung mit einem Titel versehen werden. Da sich Anmerkungen auf SISA-Ressourcen beziehen können, wird zusätzlich ein entsprechender Eintrag bereitgestellt. Abbildung 5.19 zeigt die resultierende Klasse zur Speicherung von Anmerkungen.

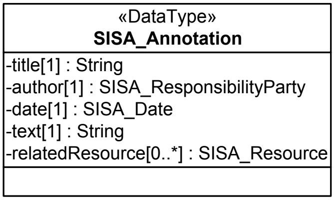


Abbildung 5.19: Datenmodell zur Speicherung von Anmerkungen. Neben dem eigentlichen Anmerkungstext (text) müssen die Einträge einen Verweis auf den Autor (author) sowie das Datum des Eintrages (date) enthalten. Der Titel (title) dient dem einfachen Zugriff auf Anmerkungen; die Zuordnung einer Anmerkung zu SISA-Ressourcen wird ebenfalls durch ein Attribut unterstützt (relatedResource).

5.2.4 Simulationslaufmanager

5.2.4.1 Komponenten-Abgrenzung

Zur Berechnung neuer Simulationsergebnisse werden vom Simulationsmodell verschiedene Daten benötigt: Systemparameter, Initialisierungsdaten, Modellumweltdaten und Optionen (vgl. Abb. 4.2, Seite 76). Die Auswahl dieser Daten wird durch einen Satz von Einstellungen im entsprechenden Simulationsmodell bestimmt. Welche Werte den einzelnen Einstellungen für einen bestimmten Simulationslauf zugeordnet werden, hängt von der Spezifikation des Simulationslaufes ab (s. Abb. 5.20).

Die Werte der einzelnen Einstellungen können sowohl Daten als auch Verweise auf Daten oder andere Ressourcen repräsentieren (z. B. die Angabe einer Jahreszahl zur Spezifizierung des Beginns eines Simulationslaufes oder ein Verweis auf einen Datensatz).

Ziele u. Funktionen

Zur Steigerung der Nachvollziehbarkeit und Reproduzierbarkeit von Simulationsläufen wurde in der Systemdefinition die Verwaltung der Simulationsläufe und Szenarien als ein Ziel definiert (/Z10/). Zu dieser Verwaltung gehören die

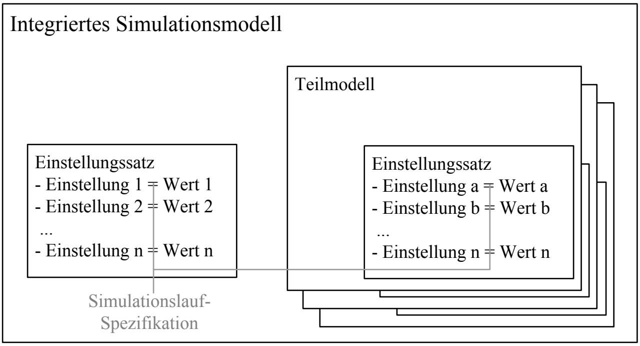


Abbildung 5.20: Simulationsmodelle und deren Einstellungen. Jedes Simulationsmodell besitzt einen eigenen Satz an Einstellungen. Die Einstellungen spezifizieren die Simulationsmodell-Daten (Systemparameter, Initialisierungsdaten, Modellumweltdaten und Optionen) und sind vom Modellbetreiber für jeden Simulationslauf festzulegen.

geforderten Funktionen zur Beschreibung von Szenarien und Simulationsläufen (/F10/ u. /F35/) und zur Daten-Zuweisung (/F20/ u. /F30/). Für die Beschreibung der Szenarien und Simulationsläufe ist bereits die Dokumentationskomponente (s. Unterabschnitt 5.2.3, Seite 119) zuständig. Die Zuordnung von Daten (allgemeiner: Ressourcen) zu Szenarien kann über die Einträge des Katalogmanagers vorgenommen werden.

Die Verantwortlichkeit der Simulationslauf-Komponente beschränkt sich damit auf die Verwaltung der Simulationslauf-Spezifikation, d. h. simulationslaufbezogener Einstellungen sowie auf die Bereitstellung dieser Informationen für die Simulationssystem-Komponente. Verantwortlichkeit

5.2.4.2 Dienst-Spezifikation

Die Dienste der Simulationslauf-Komponente können unterteilt werden in solche zur Verwaltung von Simulationsläufen (insbesondere durch den SISA-Nutzer) und solche zum lesenden Zugriff auf die Einstellungen (z. B. durch das Simulationsmodell oder durch andere Komponenten).

Die erste Schnittstelle (ISimRunManager) sollte Operationen zur Verfügung stellen, mit denen neue Simulationslauf-Spezifikationen erzeugt werden können sowie Operationen, über die Modelleinstellungen innerhalb einer Spezifikation gesetzt, geändert oder gelöscht werden können. Da sich die Modelleinstellungen zwischen den Simulationsläufen einer Simulationsstudie teilweise nur geringfügig ändern, sollte die Komponente eine Operation bereitstellen, über die eine ISimRunManager

komplette Spezifikation inkl. aller damit verbundenen Einstellungen kopiert werden kann.

ISim-Run-Specification

Der Zugriff auf die Modelleinstellungen seitens der Simulationsmodelle wird in einer separaten Schnittstelle (ISimRunSpecification) angeboten. Um sicherzustellen, dass *alle* Einstellungen, die ein Modell für einen Simulationslauf benötigt, innerhalb der Simulationslauf-Komponente vorhanden sind, sollte die Komponente eine Operation anbieten, die eine Überprüfung der Simulationslauf-Spezifikation erlaubt. Von dieser Operation sollten sowohl fehlende Einträge als auch die Überschreitung von Wertebereichen erkannt werden. Die für eine derartige Überprüfung notwendigen Informationen können entweder direkt beim Aufruf der Validierungsoperation übergeben oder über eine zusätzlich angebotene Operation bekannt gegeben werden.

Abbildung 5.21 zeigt die Schnittstellen und die ihnen zugeordneten Operationen im Überblick.

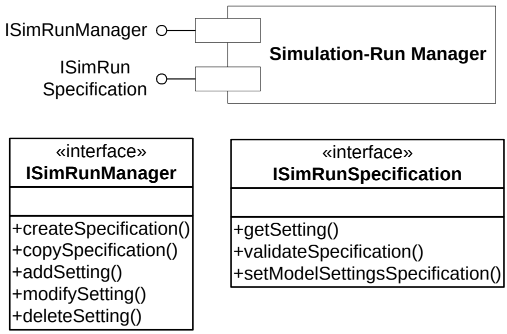


Abbildung 5.21: Schnittstellen des Simulationslaufmanagers. Der Simulationslaufmanager (Simulation-Run Manager) bietet eine Schnittstelle zur Verwaltung von Simulationslauf-Spezifikationen an (ISimRunManager) sowie eine Schnittstelle zur Abfrage der Werte von Modelleinstellungen und der Validierung von Spezifikationen (ISimRunSpecification).

5.2.4.3 Daten-Spezifikation

Abbildung 5.22 zeigt die bereits in Abb. 5.20 (Seite 129) skizzierten Zusammenhänge zwischen Simulationsmodell, Modelleinstellungen und Simulationsläufen in Form eines Klassen-Diagramms. Für die Zuordnung von Daten und Ressourcen-Referenzen zu Simulationsläufen müssen nicht alle Klassen durch entsprechende Datenstrukturen repräsentiert werden: eine Zuordnung des 'Simulationslaufes' zu einer 'Simulationslauf-Spezifikation' und die Angabe der zugehörigen 'Wertzuweisungen' reicht zur Erfüllung der Aufgabe des Simula-

tionslaufmanagers aus. Die entsprechenden Klassen hierzu finden sich in Abb. 5.23 (Seite 132). Eine Klasse zur Speicherung der Struktur von Modelleinstellungssätzen ist ebenfalls in Abb. 5.23 zu finden. Da die Speicherung und Verarbeitung dieser Strukturinformationen von System zu System sehr unterschiedlich realisiert werden kann, wird für das Spezifikationsattribut kein Datentyp angegeben.[18]

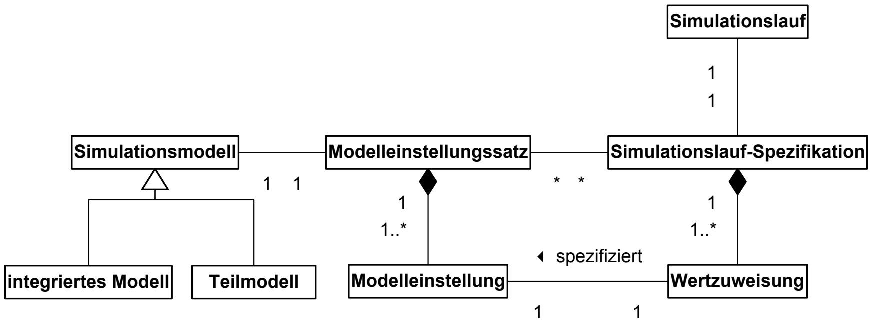


Abbildung 5.22: Zusammenhang zwischen Simulationsmodell, Modelleinstellungen und Simulationslauf. Ein Simulationsmodell besitzt einen fest definierten Modelleinstellungssatz, bestehend aus mehreren Modelleinstellungen (Einstellungsmöglichkeiten). Für jeden Simulationslauf gibt es genau eine Simulationslauf-Spezifikation, die über ihre Wertzuweisungen allen notwendigen Modelleinstellungen einen Wert zuweist. Die Zusammenhänge gelten sowohl für das integrierte Simulationsmodell als auch für die integrierten Teilmodelle.

5.2.5 Simulationssystem

5.2.5.1 Komponenten-Abgrenzung

Die zentrale Aufgabe des SISA ist die Erzeugung neuer Simulationsergebnisse (/Z60/). Die hierzu notwendigen Simulationsmodelle müssen in das Gesamtsystem integriert werden. Bei dieser Integration sollten möglichst viele Funktionen des Gesamtsystems wieder verwendet werden (Anforderung der Austauschbarkeit: /NF40/). Die in Abschnitt 3.1.2 (Seite 28) vorgestellten Systeme schlagen hierzu eine Abgrenzung des Simulationsmodells innerhalb des Gesamtsystems vor: das Prinzip der Modellierungsumgebung M (s. Seite 38) ist die klare Trennung zwischen dem mathematischen Modell, den Lösungsmethoden, den Daten,

[18] Die Strukturinformation könnte beispielsweise über XML (DTD oder XML Schema) realisiert werden. In diesem Fall wäre das Attribut als String definierbar.

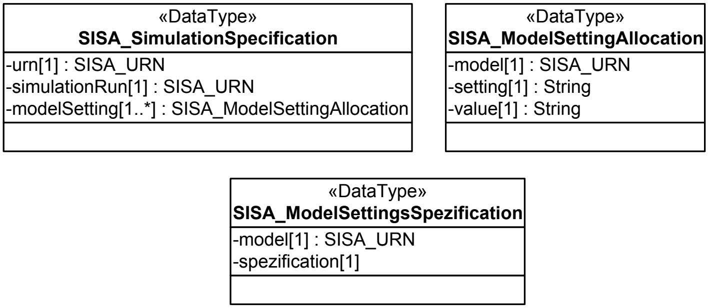


Abbildung 5.23: Datenmodell des Simulationslaufmanagers. Jedem Simulationslauf wird eine eindeutige Spezifikation zugeordnet (SISA_SimulationSpecification). Innerhalb der Spezifikation werden den Modelleinstellungen Werte zugewiesen (SISA_ModelSettingAllocation). Zur Speicherung der Struktur von Einstellungssätzen dient eine weitere Klasse (SISA_ModelSettingsSpecification).

der Datenverwaltung und der Benutzungsschnittstelle; im *Object Modeling System (OMS)* (s. Seite 31) gibt es getrennte Bibliotheken für Simulations-Module, Daten und die Visualisierung; *GLOBESIGHT* (s. Seite 29) unterscheidet zwischen der Informations-, der Funktions- und der Modellbasis. Die Erzeugung neuer Simulationsergebnisse wird daher in die Verantwortlichkeit einer Komponente gelegt: die *Simulationsmodell-Komponente*. Zur Steigerung der Wiederverwendbarkeit und Interoperabilität sollte die Komponente, über die Berechnung neuer Simulationsergebnisse hinaus, in der Lage sein, zuvor berechnete Ergebnisse über einen Dienst zur Verfügung zu stellen. Die Simulationsmodell-Komponente ist damit verantwortlich für die Berechnung, Speicherung und Weitergabe von Simulationsergebnissen.

Verantwortlichkeit

5.2.5.2 Dienst-Spezifikation

Dienste

Start

Die Erzeugung von Simulationsergebnissen integrierter Modelle erfolgt i. d. R. für Szenarien, die sich über mehrere Jahre erstrecken. Die zur Berechnung notwendigen Informationen über Modelloptionen etc. stellt der Simulationslaufmanager zur Verfügung. Für die Lieferung der Modellumweltdaten ist die Datenzugriffskomponente – in Verbindung mit dem Datenbanksystem – verantwortlich. Alle zum Start eines Simulationslaufes notwendigen Informationen und Daten sollten vor dem Start einer Simulation in einer Simulationslauf-Spezifikation (über den Simulationslaufmanager) festgelegt sein, so dass die Simulationssys-

temkomponente mit dem Aufruf einer Operation (run) die Simulation beginnen kann.

Initialisierung

Zur Berechnung der Modellergebnisse für das erste Jahr eines Simulationszeitraums sind i. d. R. Anfangswerte innerhalb der Simulationsmodelle zu belegen: Optionen müssen gesetzt/eingelesen und Startwerte von Modellvariablen gesetzt oder berechnet werden, Teilmodelle müssen Kontakt mit anderen Teilmodellen oder dem Gesamtmodell aufnehmen etc. Die Phase dieser *Initialisierung* eines Simulationssystems wird im Allgemeinen von der Berechnung der Simulationsergebnisse getrennt. Die Modellkomponenten von OMS (vgl. Unterabschnitt 3.1.3, Seite 31) besitzen stets die drei Methoden *register* (zur Registrierung von Modellteilen), *init* (zur Initialisierung) und *run* (zum Aufruf der eigentlichen Funktionalität). Auch die HLA (vgl. Unterabschnitt 3.2.2, Seite 47) sieht einen eigenen Zustand (initialization) für die Initialisierung eines Modellteils vor (IEEE, 2000b).

Vorteil

Die Aufteilung in eine Initialisierungs- und eine Berechnungsphase hat den Vorteil, dass das Simulationssystem Szenarien schrittweise, also in vorgegebenen Intervallen, berechnen kann und dadurch Interventionen durch Dritte (Modellnutzer oder Software) möglich werden.

Nachteil

Auch wenn die Unterteilung in eine Initialisierungs- und eine Berechnungsphase den Nachteil mit sich bringt, dass die Komponente *zustandsbehaftet* ist[19], sollte sie wegen des angeführten Vorteils eine entsprechende Operation zur Initialisierung eines Simulationslauf (init) anbieten. Da die für einen Simulationslauf notwendigen Daten vom Simulationslaufmanager verwaltet werden, sollten zur Initialisierung die dort eingeführten eindeutigen Simulationslaufnamen verwendet werden.

Ergebnisabfrage

Um die Interoperabilität zwischen verschiedenen SISAs zu erhöhen und den Zugriff auf Simulationsergebnisse von anderer Software zu vereinfachen, sollte die Simulationskomponente selbst eine Schnittstelle zur Abfrage von Simulationsergebnissen bereitstellen.

Abbildung 5.24 zeigt die resultierenden Schnittstellen und Operationen der Simulationssystemkomponente in der Übersicht. Dynamische Aspekte der Komponente sind in Abb. 5.37 (Seite 155) zusammengefasst.

5.2.5.3 Daten-Spezifikation

Für das Simulationssystem wird keine Daten-Spezifikation vorgenommen. Für konkrete Systeme müssen für diese Komponente die Klassen für mögliche Ergebnisrückgaben definiert werden, die die Resultate der Operation *getResult()* bestimmen.

[19] Was nach der *simple service architecture* zu vermeiden ist (vgl. Seite 58).

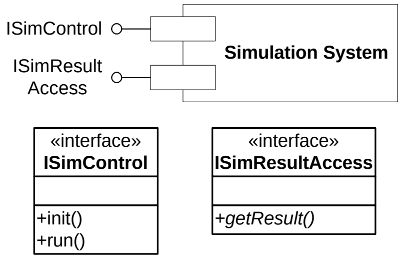


Abbildung 5.24: Schnittstellen des Simulationssystems. Die Operationen der Schnittstelle zur Simulationskontrolle (ISimControl) dienen der Initialisierung (init) und dem Start (run) von Simulationsläufen. Die Schnittstelle zum Ergebniszugriff (ISimResultAccess) besitzt lediglich eine abstrakte Operation, die für die Rückgabe von Simulationsergebnissen verantwortlich ist (getResult).

5.2.6 Datenzugriff und Datenbanksystem

5.2.6.1 Komponenten-Abgrenzung

Für ein integriertes simulationsbasiertes Assessment ist die Verwendung vielfältiger Datensätze notwendig (vgl. Abb. 4.2, Seite 76 und Unterabschnitt 4.2.5, Seite 84): Daten zur Initialisierung und zum Betrieb der Simulationsmodelle, Primärdaten zur Erzeugung dieser Eingabedaten, Hintergrunddaten für weitere Analysen usw. Die Daten werden demnach nicht nur vom integrierten Simulationsmodell verwendet, sondern auch von anderen Komponenten des SISA (beispielsweise den Komponenten zur Datenvorverarbeitung und Ergebnisanalyse).

Integration

Ein Ziel des SISA ist die Integration von Daten (/Z70/). Ziel einer solchen *Datenintegration* ist es nach Thomas und Nejmeh (1992), sicherzustellen, dass alle Daten innerhalb des Systems als ein *konsistentes Ganzes* verwaltet werden, unabhängig davon, wie auf die Teile des Ganzen eingewirkt wird. In diesem Zusammenhang stellen sich zwei Fragen bezüglich der Datenhaltung:

2 Fragen

- Wo (in welcher Komponente/an welchem Ort) werden die Daten gehalten?
- Wie (in welchem Format) werden die Daten gespeichert?

Ort

Bei der Frage nach dem Speicherort von Daten gibt es zwei grundsätzliche Alternativen: die Datensätze werden in derjenigen Komponente verwaltet, in der sie (am meisten) benötigt werden oder in einer zentralen Datenhaltungskomponente (Datenbanksystem[20]). Gegen die dezentrale Datenhaltung spricht

[20]Siehe Erklärung des Begriffs im Glossar, Seite 229.

der bereits angesprochene Umstand, dass dieselben Daten in verschiedenen Komponenten benötigt werden und damit jede an einem Datenaustausch beteiligte Komponente entsprechende Schnittstellen zum Datenaustausch mit den anderen Komponenten zur Verfügung stellen müsste. Ein weiterer Grund gegen die dezentrale Datenhaltung ist der hohe Aufwand, der mit einer Implementierung der Funktionen zur Datenverwaltung verbunden ist. Bei einer zentralen Datenverwaltung benötigt jede Komponente hingegen nur eine Schnittstelle zum Datenaustausch. Darüber hinaus fördert die zentrale Datenhaltung die Sicherstellung der Konsistenz der eingesetzten Daten und erlaubt die einfachere Wiederverwendung von Datensätzen beim Austausch des integrierten Modells. Gegen die zentrale Datenhaltung sprechen evtl. notwendige Änderungen in den zugreifenden Komponenten und evtl. auftretende Geschwindigkeitseinbußen bei der Datenübertragung.

Format

Die Speicherung aller Daten in einem einheitlichen Format wäre aus verwaltungstechnischer Sicht und aus Gründen der Interoperabilität die beste Lösung. Die Verwendung eines einheitlichen Formats ist aber nicht immer realisierbar, da die Teilmodelle innerhalb eines integrierten Modells i. d. R. unterschiedliche Datenformate verwenden – sowohl zur persistenten Datenspeicherung (z. B. in Form von Textdateien, Datenbanktabellen oder Binärdateien) als auch zur internen Repräsentation von Daten (z. B. in ein- oder zweidimensionalen Arrays, Containern oder Listen). Die gleichen Anmerkungen gelten für Software-Werkzeuge von Drittanbietern, deren Zugriffsoperationen i. d. R. überhaupt nicht geändert werden können, womit die Datenformate zwingend vorgeschrieben sind.

Folgerung

Das SISA sollte demnach eine zentrale Datenhaltung unter Verwendung einheitlicher Datenformate anbieten, gleichzeitig aber offen sein gegenüber einer dezentralen Datenspeicherung unterschiedlicher (nicht vorbestimmter) Datenformate. Aus Gründen der Interoperabilität und der Austauschbarkeit von Daten sollte der Datenzugriff auf die zentral und dezentral gespeicherten Daten transparent sein, d. h. die zugreifenden Komponenten sollten weder ‘wissen müssen’ wo sich die angeforderten Daten befinden, noch in welchem Format diese Daten gespeichert sind.

Prinzip

Aus diesen Überlegungen ergibt sich die in Abb. 5.25 (Seite 136) dargestellte Strukturierung zur Datenhaltung und zum Datenzugriff: wenn Komponenten, wie beispielsweise die Simulationskomponente, Daten benötigen, so greifen sie über die *Datenzugriffskomponente* auf die Daten in der Datenbasis zu. Die Transformationen von Datenformaten (vgl. die Ausführungen zum Katalogmanager in Unterabschnitt 5.2.1, Seite 96) wird ebenfalls von dieser Komponente angeboten. Innerhalb der Datenbasis sollten möglichst alle Daten im *Datenbanksystem* gespeichert sein. Zur Verwaltung der Daten innerhalb des Datenbanksystems bietet die Komponente, die z. B. durch ein eigenständiges Datenbank-Managementsystem realisiert werden kann, entsprechende Funktio-

nen an. Daten, die sich aus technischen Gründen nicht im Datenbanksystem integrieren lassen, werden ebenfalls von der Datenzugriffskomponente zur Verfügung gestellt. Die Informationen, die für einen Zugriff auf die Daten notwendig sind, werden vom Katalogmanager zur Verfügung gestellt. Lassen sich die Operationen zum Datenzugriff innerhalb einer Komponente (z. B. in einem Teilmodell) nicht ändern, kann sie – an der Datenzugriffskomponente vorbei – direkt auf die Daten zugreifen. Die Integration dieser Daten geschieht dann lediglich über die zugehörigen und im Katalogmanager verwalteten Metadaten.

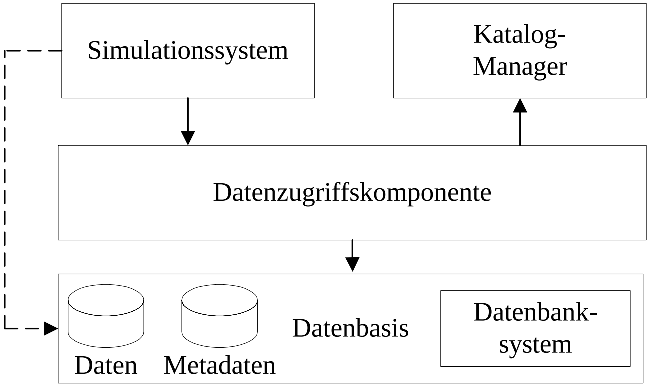


Abbildung 5.25: Komponenten zum Datenzugriff und zur Datenhaltung (Datenbanksystem). Erklärungen finden sich im Text.

Verantwortlichkeiten

Das *Datenbanksystem* ist damit für die Verwaltung und persistente Speicherung von Daten verantwortlich. Die *Datenzugriffskomponente* ist zuständig für die transparente Bereitstellung von Daten. Die *Datenbasis* ist keine Komponente im eigentlichen Sinne; sie kann als eine Ansammlung von Daten gesehen werden.

SISA-Ziele

Das *Datenbanksystem* trägt somit zur *Datenintegration* (/Z70/) und *Ergebnisbereitstellung* (/Z90/) bei. Die der *Datenzugriffskomponente* zugeordneten Dienste zur Transformation von Daten können nicht nur von den Simulationsmodellen zur Laufzeit verwendet werden: der *Datenimport* (/F150/) und *Datenexport* (/F170/) kann ebenfalls über diese Komponente abgewickelt werden. Sofern die Komponente auch Dienste zur Transformation der Daten in Formate bereitstellt, die einfach zur Visualisierung benutzt werden können (z. B. die Abbildung von Rasterdaten in Bilder), wird damit ebenfalls das Ziel der *Ergebnisbereitstellung* unterstützt (/Z40/ und /F160/). Darüber hinaus tragen *beide Komponenten* in hohem Maße zu den geforderten nicht-funktionalen Anforderungen der *Interoperabilität* (/ZN10/) und *Austauschbarkeit* (/ZN40/) bei.

5.2.6.2 Dienst-Spezifikation

Datenbanksystem

In Anlehnung an die vom OpenGIS-Konsortiums vorgeschlagenen und bereits im Unterabschnitt des Katalogmanagers aufgeführten primären Funktionen zum Datenzugriff (vgl. Tab. 5.2, Seite 103), enthält die Schnittstelle des Datenbanksystems (IDBSystem) Operationen, um neue Datensätze zu erzeugen (addDataset), abzufragen (retrieve), zu modifizieren (modify) und zu entfernen (removeDataset). Um einen gleichzeitigen Zugriff mehrerer Komponenten (oder Teilmodelle) auf die Datensätze zu erlauben, werden die in Nebert (2002) aufgeführten Funktionen zur Abfrage des Zugriffsstatus (check/modify/get-DatabaseStatus) ebenfalls in die Schnittstelle aufgenommen (s. Abb. 5.27). Damit ein Metadaten-Sammler auch Zugriff auf die Metadaten hat, die direkt im Datenbanksystem gespeichert sind, bietet die Komponente über eine zusätzliche Schnittstelle (IDBDiscovery) eine weiter Operation zur Abfrage von Datensätzen (query).

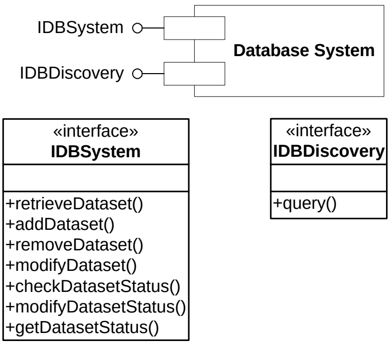


Abbildung 5.26: Schnittstelle des Datenbanksystems. Erklärungen finden sich im Text.

Datenzugriffskomponente

Die Datenzugriffskomponente kann als Middleware zwischen der Datenbasis und den anderen Komponenten angesehen werden (vgl. Abb. 5.5, Seite 104). Zur Gewährleistung eines transparenten Datenzugriffs bietet sie daher, Kottmann (1999c) folgend, die gleiche Schnittstelle zur Datenhaltung an wie das Datenbanksystem (IDBSystem). Die Operation zur Format-Transformation eines Datensatzes (transformDataset) wird in einer zusätzlichen Schnittstelle (IDataTransform) angeboten (s. Abb. 5.27).

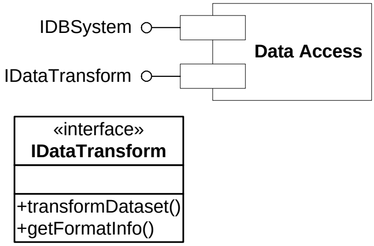


Abbildung 5.27: Schnittstelle der Datenzugriffskomponente. Erklärungen finden sich im Text.

5.2.6.3 Daten-Spezifikation

Datenhaltungskomponente

Das Datenmodell des Datenbanksystems ist abhängig von der Realisierung der Komponente.

Datenzugriffskomponente

Die Datenzugriffskomponente benötigt zur Realisierung ihrer Schnittstelle zumindest ein entsprechendes Attribut zur Beschreibung der unterstützten Transformationsformate (formatInfo, s. Abb. 5.28).

Die unterstützten Datenformate sollten sich aus Gründen der Interoperabilität allerdings an Standards orientieren. Als wichtige, vielerorts eingesetzte Standards sind die Implementierungs-Spezifikationen des OGC zu den *Simple Features* zu betrachten (Kottmann, 1999e).

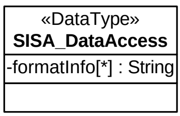


Abbildung 5.28: Datenmodell der Datenzugriffskomponente. Erklärung findet sich im Text.

5.2.7 Geodatenverarbeitung

5.2.7.1 Komponenten-Abgrenzung

Die georäumliche Auflösung ist wichtig, da sie eine verbesserte Repräsentation globaler dynamischer *Prozesse* (inkl. Rückkopplungen) erlaubt und detaillierte-

re Informationen für Auswirkungs-Analysen ermöglicht (Alcamo u. a., 1998a). Auf die Wichtigkeit georäumlicher *Daten* und die zunehmende Integration von GIS bzw. GIS-Funktionalität in simulationsbasierte Systeme wurde bereits in der Systemdefinition hingewiesen. Aktuelle Entwicklungen berücksichtigen diese Anforderung: *OMS* (s. Seite 31) durch die Bereitstellung eines GIS-Clients zur Bearbeitung und Visualisierung georäumlicher Daten und das System von Villa und Costanza (2000) durch eine GIS-Komponente als zentrales System-Element.

Integration

Die Integration typischer GIS-Funktionen[21] kann auf zwei verschiedene Arten erfolgen: einerseits durch eine direkte Implementierung der geforderten Funktionen durch das SISA und andererseits durch die Anbindung eines eigenständigen (evtl. kommerziellen) GIS.

Transparenz

Welche der beiden Möglichkeiten in einem System verwendet wird hängt u. a. von Komplexitäts- und Effizienzfragen ab. Der Aufwand zur Implementierung einer sehr komplexen Funktion innerhalb des SISA kann zu groß sein, so dass nur eine Anbindung an ein eigenständiges GIS in Frage kommt. Zur Steigerung der Ausführungsgeschwindigkeit kann es aber sinnvoll sein, einfache Funktionen direkt im SISA zu realisieren. Welche der beiden Möglichkeiten in einem System verwendet wird, sollte für den Aufrufer einer Funktion allerdings irrelevant (und der Aufruf damit *transparent*) sein.

Verantwortlichkeit

Zur Steigerung der Transparenz und Wiederverwendbarkeit sowie zur besseren Austauschbarkeit GIS-typischer Funktionen, sollten diese Funktionen über eine weitere Indirektion in Form einer Komponente zur *Geodaten-Verarbeitung* angeboten werden. Diese Komponente ist verantwortlich für die Verarbeitung geographischer Daten und die Bereitstellung einer Schnittstelle zu eigenständigen GIS. Sie liefert damit einen Beitrag zur Vorverarbeitung und Nachbearbeitung von Simulationsmodell-Daten (/Z80/) sowie zu den nicht-funktionalen Zielen der Interoperabilität (/ZN10/), Modifizierbarkeit (/ZN30)/) und Austauschbarkeit (/ZN40/).

Abgrenzung

Da im Umfeld eines SISA i. d. R. eigenständige GIS verwendet werden, beschränkt sich die Verantwortlichkeit der Komponente bewusst auf das Anbieten von Funktionen zur *Datenverarbeitung* und übernimmt keine weiteren GIS-Aufgaben wie z. B. die Visualisierung oder Datenhaltung.

5.2.7.2 Dienst-Spezifikation

Die geographischen Komponenten von Modellen zum integrierten Assessment arbeiten meist mit Rasterdaten (vgl. Bakkes u. a., 2000). Die Komponente der Geodatenverarbeitung sollte daher zumindest typische Raster-GIS-Funktionen zur Verfügung stellen. Zu den elementaren Operationen der Rasterdatenverarbeitung gehören nach Bartelme (2000) die *radiometrischen Transformationen*

[21] Beispielsweise die Verschneidung von Datenschichten (Layers), die Pufferbildung und die Reklassifizierung.

sowie *arithmetische und logische Kombinationen* von Rasterbildern. Bei radiometrischen Transformationen wird eine Transferfunktion auf die Werte (Grauwerte) der Rasterzellen angewendet (z. B. eine Multiplikation aller Werte mit einer Konstanten oder die Anwendung von Schwellwerten zur Unterdrückung oder Hervorhebung bestimmter Zellen). Bei arithmetischen und logischen Operationen werden die Grauwerte von zwei Rasterbilder miteinander kombiniert (z. B. addiert, multipliziert oder UND-verknüpft).

ISO/DIS 19119

In der Dienste-Taxonomie der *OpenGIS Service Architecture (ISO/DIS 19119)* (vgl. Abschnitt 3.2.5, Seite 55; Percivall, 2002) werden die geographischen Verarbeitungsdienste in Dienste eingeteilt, die sich auf den Raum, das Thema, die Zeit oder die Metadaten beziehen.

Raum

Dienste, die den Raumbezug von Daten ändern, werden (wenn überhaupt) vornehmlich zur Datenvorverarbeitung benötigt – Beispiele sind die Koordinatentransformation, die Änderung der Größe von Rasterzellen oder die Rasterung von Vektordaten.

Thema

Zu den Diensten zur Änderung thematischer Aspekte gehören die oben angesprochenen *elementaren Operationen*, die in der Taxonomie unter dem Begriff *geographic calculation service*[22] zusammengefasst sind.

Zeit

Die Dienste zur Verarbeitung von Geodaten, die zeitliche Aspekte berücksichtigen, umfassen sowohl Operationen zur Transformation zwischen zeitlichen Referenzsystemen als auch Operationen, um aus Zeitreihen Stichproben zu nehmen sowie Funktionen zur Datenselektion aufgrund von Zeitpunkten oder Zeitintervallen.

Metadaten

Zu den Diensten, die sich auf die Metadaten von geographischen Daten beziehen, gehört der *statistical calculation service*. Dieser Dienst ist zur statistischen Auswertung von Datensätzen gedacht, dessen Operationen beispielsweise den Mittelwert, den Modalwert oder die Standardabweichung eines Datensatzes berechnen oder ein Histogramm erstellen. Ein solcher Dienst kann daher als Ausgangspunkt für die Realisierung der gewünschten SISA-Funktion zur statistischen Auswertung (/F180/) genutzt werden.

Auswahl

Die zur Taxonomie aufgeführten Dienste sollen laut ISO/DIS 19119 lediglich als Beispiele gesehen werden. Einige der angeführten Dienste können auch im Rahmen eines SISA sehr gut eingesetzt werden (die oben genannten Dienste sind ebenfalls nur Beispiele). Eine Auswahl und Spezifikation der einzelnen Dienste ist an dieser Stelle allerdings wenig sinnvoll, da sie vom konkret zu realisierenden SISA abhängt. Die Schnittstellendefinition zeigt daher lediglich die beiden Operationen zur Transformation (transformGrid) und Kombination (combineGrid) von Rasterdatensätzen und eine Funktion zur statistischen Analyse eines Rasterdatensatzes (gridStatistics). Abbildung 5.29 fasst die Schnittstellen und Operationen der Komponente zusammen.

[22] „Dienste zur Ableitung anwendungsorientierter, quantitativer Ergebnisse, die nicht von den Rohdaten selbst verfügbar sind.“ Percivall (2002)

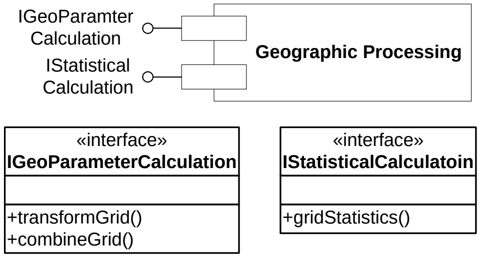


Abbildung 5.29: Schnittstellen der Komponente zur Geodatenverarbeitung.

5.2.7.3 Daten-Spezifikation

Die Spezifikation der unterstützten Datenmodelle sollte sich an den eingesetzten Klassen der Datenzugriffskomponente orientieren und wird an dieser Stelle nicht weiter ausgebaut.

5.2.8 Datenverarbeitung

5.2.8.1 Komponenten-Abgrenzung

In einem SISA gibt es Funktionen, die unabhängig von einem konkreten Projekt realisiert und innerhalb verschiedener Systeme eingesetzt werden können; die im vorigen Unterabschnitt (5.2.7, Seite 138) erwähnten Dienste zur Verarbeitung von Geodaten sind wichtige Vertreter dieser Funktionen. Neben den Funktionen zur Bearbeitung geographischer Daten gibt es weitere Funktionen, die im Rahmen eines SISA oft benötigt werden. Als Beispiele seien die Erzeugung von Zufallszahlen, die bei Simulationen oft eine wichtige Rolle spielt (s. z. B. Bratley u. a., 1987; Grams, 1992; Steinhausen, 1994) oder die Interpolation und numerische Integration von Werten (s. z. B. Bossel, 1994; Grams, 1992; Liebl, 1995) genannt. Die Berechnung statistischer Daten (Minimal-, Maximal-, Mittelwert, Standardabweichung etc.) fällt ebenfalls in die Kategorie oft benötigter Funktionen (insbesondere zur Ergebnisanalyse).

Zusammen mit den Diensten zur Geodatenverarbeitung können derartige Funktionen als ‘shared processing services’ im Sinne der Dienste-Architektur des OpenGIS (ISO/DIS 19119) angesehen werden (vgl. Unterabschnitt 3.2.5, Seite 55), die verantwortlich für die Bereitstellung von allgemeinen, durch mehrere Nutzer verwendbaren Funktionen sind (s. Percivall, 2002). Die „Shared Domain Services“, zu denen auch die Geodienste gehören, können als „toolbox“ oder „building blocks of Applications“ angesehen werden (Kottman, 1999). In diesem Sinne ist die Datenverarbeitungskomponente, die im Folgenden auch als ‘Utility’-Komponente bezeichnet wird, verantwortlich für die Bereitstel- Shared Services Verantwortlichkeit

lung allgemeiner Datenverarbeitungsdienste. Die Komponente dient im SISA der Unterstützung bei der Erzeugung von Simulationsergebnissen (/Z60/), der Datenvorverarbeitung und Nachbearbeitung (/Z80/) und der Ergebnisanalyse (/Z100/).

5.2.8.2 Dienst-Spezifikation

Die Funktionen der Komponente sind abhängig von den Anforderungen eines konkret zu realisierenden SISA, so dass auf eine detaillierte Spezifikation an dieser Stelle verzichtet wird. Die Schnittstelle der Komponente (IUtility) enthält daher lediglich eine exemplarische Operation zur Berechnung statistischer Größen.

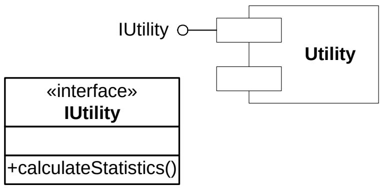


Abbildung 5.30: Schnittstelle zur allgemeinen Datenverarbeitung. Eine Spezifikation der Schnittstelle (IUtility) findet an dieser Stelle nicht statt.

5.2.8.3 Daten-Spezifikation

Das Datenmodell der Dienste sollte sich – wie bereits das Modell der Komponente zur Geodatenverarbeitung – an den eingesetzten Klassen der Datenzugriffskomponente orientieren und, soweit möglich, Standards berücksichtigen.

5.2.9 Aufgabensteuerung

5.2.9.1 Komponenten-Abgrenzung

Die einzelnen Komponenten der SISA-Architektur stellen in ihren Schnittstellen grundsätzlich wieder verwendbare Operationen bereit. Die Operationen der Komponente zur Geodatenverarbeitung stehen beispielsweise nicht nur dem Simulationssystem zur Verfügung, sondern auch allen anderen Komponenten und anderen Systemen. Bei der Vorverarbeitung und Nachbearbeitung von Daten können SISA-Betreiber und Modellentwickler daher ebenfalls diese Dienste in Anspruch nehmen. Um die Nutzung der Dienste zu vereinfachen, sollte das SISA daher Funktionen zum Aufruf und zur Kontrolle von Diensten bereitstellen.

Die Aufgabensteuerung ist somit verantwortlich für den programmgesteuerten Aufruf anderer Dienste des SISA. Verantwortlichkeit

5.2.9.2 Dienst-Spezifikation

Dienstekette

Eine Vielzahl von Aufgaben wird nicht durch den Aufruf nur *eines* Dienstes zu lösen sein (zur Konvertierung eines Datensatzes kann beispielsweise eine räumliche und eine thematische Verarbeitung notwendig sein). Dienste sollten also verkettet und als Ganzes aufgerufen werden können. In der ISO/DIS 19119 (Percivall, 2002) wird bei der Verkettung von Diensten (vgl. Seite 57, Unterabschnitt 3.2.5) unterschieden zwischen *nutzerdefinierten* Ketten, bei denen der Nutzer den Arbeitsablauf innerhalb der Kette selbst definiert und kontrolliert, *Workflow-verwalteten* Ketten, die vom Nutzer über einen Verwaltungsdienst aufgerufen werden, der dann die einzelnen Dienste der Kette kontrolliert und *aggregierten Diensten*, bei denen der Nutzer eine Kette als einfachen Dienst aufruft und sich nicht darüber bewusst ist, dass es sich um eine Dienstekette handelt.

Das von der ISO/DIS 19119 vorgestellte Konzept zur Verkettung von Diensten sieht einen umfangreichen Dienstekatalog vor. In diesem Katalog sind beispielsweise Dienste zur Definition, Kontrolle und zur Abfrage des Status von Diensteketten zu finden, Dienste zur Anzeige/Verwaltung von Metadaten über Dienste, Dienste zur Validierung von Diensteketten sowie Dienste zur Autorisierung und Authentifizierung. Zur Verkettung von Diensten müssen die Dienste selbst einige Voraussetzungen erfüllen: sie müssen genau definierte Schnittstellen aufweisen, müssen miteinander verbunden und mit Daten gekoppelt werden.

Kontrollstrukturen

Über die reine Zusammensetzung von Diensten zu Diensteketten hinaus wäre es sinnvoll, wenn die Komponente einfache Steuerungsmechanismen (Kontrollstrukturen) bereitstellen würden. Beim *Object Modeling System* (vgl. Unterabschnitt 3.1.3, Seite 31) wird diese Funktionalität beispielsweise über die Integration eines Skript-Interpreters realisiert.

abstrakte Operationen

Da die dynamische Zusammenstellung von Diensteketten ausdrücklich nicht zu den Hauptaufgaben des SISA gehört (s. Systemdefinition, Seite 80), wird an dieser Stelle keine detaillierte Spezifikation der Komponenten-Schnittstellen vorgenommen. Die Umsetzung des gesamten Dienstekonzepts der ISO/DIS 19119 wird im Rahmen einer SISA-Entwicklung nur selten realisierbar sein. Das Grundkonzept der dokumentierten Zusammenstellung vorhandener Dienste sollte dennoch in die Architektur einfließen. Der Komponente werden daher zwei *abstrakte*, für eine konkrete Realisierung weiter auszuführende Operationen zugeordnet (s. Abb. 5.31, Seite 144): eine für die Definition von Aufgaben (defineTask) und eine zur aggregierten Ausführung vordefinierter Aufgaben (invokeTask).

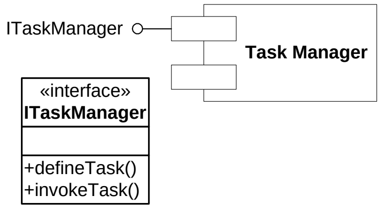


Abbildung 5.31: Schnittstelle zur Aufgabensteuerung. Die Komponente stellt lediglich zwei abstrakte Operationen bereit, um Aufgaben zu definieren (defineTask) und durchzuführen (invokeTask).

5.2.9.3 Daten-Spezifikation

Die Metadaten für Dienste sollten in den Katalogmanager aufgenommen werden.[23] Informationen über die Verwendbarkeit von Diensten zur Bearbeitung häufig wiederkehrender Funktionen sollten im *Service Organizer Folder* aufgenommen werden. Weitere Spezifikationen werden an dieser Stelle nicht vorgenommen.

5.2.10 Ergebnisanalyse

5.2.10.1 Komponenten-Abgrenzung

Das SISA soll, neben der Erzeugung neuer Simulationsergebnisse (/Z60/), auch Funktionen vorhalten, die bei der Bereitstellung (/Z90/) und Analyse (/Z100/) der Simulationsergebnisse helfen. Einige der vorgestellten Komponenten des SISA unterstützen diese Ziele bereits: bei den Komponenten für die geographische bzw. allgemeine Datenverarbeitung wird beispielsweise eine Operation zur Berechnung statistischer Werte vorgeschlagen (vgl. Unterabschnitt 5.2.7, Seite 138 bzw. 5.2.8, Seite 141) und bei der Datenzugriffskomponente wird die Transformation in Datenformate vorgeschlagen, die eine Visualisierung der Daten erleichtert.

Verantwortlichkeit

Aufbauend auf den Datenverarbeitungsdiensten und den Diensten der Datenzugriffskomponente ist die Analysekomponente verantwortlich für die Unterstützung des Modellbetreibers bei der Analyse von Simulationsergebnissen.

5.2.10.2 Dienst-Spezifikation

Alle vom Modellbetreiber zur Analyse eines Problems gewünschten Dienste und Operationen sollten über diese Komponente angeboten werden. In Anlehnung

[23] ISO/DIS 19119 stellt hierzu ein eigenes Metadaten-Schema zur Verfügung.

an die geforderten Funktionen zur Visualisierung (/F160/) und statistischen Auswertung von Daten (/F180/) enthält die Schnittstelle der Analysekomponente (IAnalysis) zunächst zwei entsprechende Operationen (s. Abb. 5.32). Da die Operationen der Analysekomponente von den konkreten Anforderungen eines SISA abhängig sind, werden über die beiden angegebenen Operationen hinaus an dieser Stelle keine weiteren Operationen spezifiziert.

Um die Analyse von Daten zu erleichtern, sollten für alle Assessment-Daten des SISA (bzw. deren Formate/Klassen) entsprechende Visualisierungsoperationen bereitgestellt werden, wobei die Operationen in der Lage sein sollten, Daten in unterschiedlichen Formen (z. B. als Karte, Diagramm oder Tabelle) darzustellen. Darstellungformen

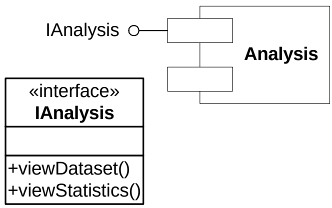


Abbildung 5.32: Schnittstelle der Analysekomponente. Die Analysekomponente enthält zwei exemplarische Operationen: eine zur Visualisierung von Datensätzen (viewDataset) und eine zur Visualisierung statistischer Werte eines Datensatzes (viewStatistics). Da die Komponente verantwortlich ist für die *Bereitstellung* von Simulationsergebnissen für den *Modellbetreiber*, beschränken sich die Operationen auf die *Visualisierung* der entsprechenden Daten.

5.2.10.3 Daten-Spezifikation

Für die Analysekomponente werden keine gesonderten Daten-Spezifikationen vorgenommen. Die von den Diensten unterstützten Datenmodelle sollten sich allerdings an den eingesetzten Klassen der Datenzugriffskomponente orientieren, um eine nahtlose Datenverarbeitung zu ermöglichen.

5.2.11 Modellanalyse

5.2.11.1 Komponenten-Abgrenzung

Die Wichtigkeit der Durchführung von Unsicherheits- und Sensitivitätsanalysen für die verwendeten Simulationsmodelle wurde bereits in der Systemdefinition (Kapitel 4, Seite 69) herausgestellt. In diesen Analysen werden gezielte Veränderungen der Eingangsdaten der Simulationsmodelle vorgenom- Modellanalyse

men und in Beziehung zur Änderung der Ausgangsgrößen gesetzt. Die Trennung der Verantwortlichkeiten für die Erzeugung von Simulationsergebnissen (Simulationsmodell-Komponente) und die Verwaltung bzw. Bereitstellung von Modelleingabedaten (Simulationslaufmanager-Komponente) erlaubt eine einfache Änderung der Eingabedaten (z. B. die Änderung von Modellparametern). Diese Änderungen sollten von einer separaten Komponente, der *Modellanalyse-Komponente*, durchgeführt werden. Die *Modellanalyse-Komponente* ist damit verantwortlich für die Unterstützung bei der Analyse des Simulationssystems, insbesondere bei der Durchführung von Sensitivitäts- und Unsicherheitsanalysen.

Verantwortlichkeit

5.2.11.2 Dienst-Spezifikation

Unsicherheiten

Die Modellumweltgrößen sowie die Parameterwerte von Modellen können i. d. R. nicht genau angegeben werden. Die Unsicherheiten dieser Werte müssen daher stets berücksichtigt werden. Im Zusammenhang mit den Eingabedaten eines Modells können Unsicherheiten zwei Ursachen haben: Zum einen Unsicherheiten bei den Messungen und zum anderen inhärente stochastische Schwankungen bei den gemessenen Werten. Diese Unsicherheiten können z. B. berücksichtigt werden, indem die Eingabewerte zufällig im Rahmen der Unsicherheit des Parameters verändert werden (s. z. B. Clark u. a., 1975).

gezielte Wertänderungen

Das Simulationsmodell bekommt seine Eingabedaten von der Datenzugriffskomponente. Zur Analyse der Sensitivität und Unsicherheit sind die Eingabedaten allerdings gezielt zu verändern. Diese Veränderungen könnten direkt *innerhalb* des Simulationssystems vorgenommen werden. Hierzu müsste das Simulationssystem entsprechende Methoden bereitstellen und die Eingabedaten dann gezielt (z. B. über interne Faktoren) manipulieren. Ein anderer Weg ist die Veränderung der Eingabedaten *bevor* sie dem Simulationssystem übermittelt werden. Diese Art der Veränderung von Eingabewerten hat den Vorteil, dass die Modelle selbst keine entsprechenden Vorkehrungen zur Manipulation der Daten treffen müssen. Da die Wahl der zu verändernden Parameter und die Kombinationen bei der gleichzeitigen Veränderung mehrerer Parameter eine nicht-triviale Aufgabe ist, sollte diese Verantwortlichkeit in eine separate Komponente gelegt werden.

Zwischenschicht

Um ihre Aufgabe der Manipulation von Eingangsdaten erfüllen zu können, befindet sich die Modellanalyse-Komponente als Schicht zwischen Simulationssystem und Datenzugriffskomponente (s. Abb. 5.33). Alle Daten, die das Simulationssystem während eines Simulationslaufes (Analyselaufes) benötigt, werden von der Modellanalyse-Komponente zur Verfügung gestellt; ein direkter Zugriff des Simulationssystems auf die Datenzugriffskomponente findet während eines solchen Analyselaufes nicht statt. Auf diese Weise können alle Eingabedaten vor der Weitergabe an das Simulationssystem gezielt verändert werden.

Die zur Modellanalyse benötigten Ausgabedaten des Simulationssystems gehen ebenfalls den Weg über die Modellanalyse-Komponente.

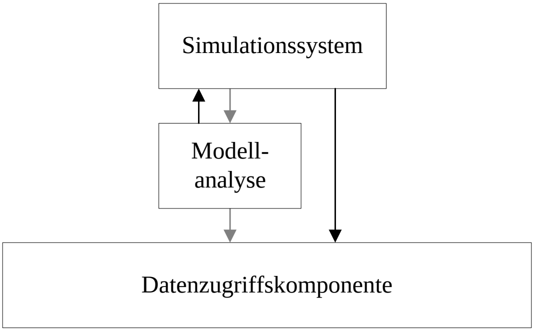


Abbildung 5.33: Prinzip der Modellanalyse. Die Modellanalyse-Komponente wird als Schicht zwischen das Simulationssystem und die Datenzugriffskomponente gelegt. Alle Modelleingabedaten und Modellausgabedaten passieren die Modellanalyse-Komponente und können auf diese Weise verändert und ausgewertet werden.

Für eine Modellanalyse sind i. d. R. eine ganze Reihe an Simulationsläufen durchzuführen. Die grundlegenden Einstellungen für die Simulationsläufe sind wiederum über eine Simulationslauf-Spezifikation, also über den Simulationslaufmanager, festzulegen. Die speziell für eine Modellanalyse benötigten Parameter (z. B. die maximale Veränderung von Eingabedaten) sind direkt von der Modellanalyse-Komponente vorzuhalten.

Die Modellanalyse-Komponente muss verschiedene Schnittstellen implementieren. Da der Datenzugriff für das Simulationssystem transparent sein sollte, muss die Modellanalyse-Komponente zunächst die gleichen Schnittstellen bereitstellen wie die Datenzugriffskomponente (IRepositor und IDataAccess). Darüber hinaus muss sie eine Schnittstelle bereitstellen, die die Analyseoperationen – zumindest eine abstrakte Operation zum Start der Analyse (analyse) – aufnimmt (ISimModelAnalysis). Die Schnittstellen der Komponente sind in Abb. 5.34 (Seite 148) zusammengefasst.

Schnittstellen

5.2.11.3 Daten-Spezifikation

Die für die Durchführung von Modellanalysen notwendigen Daten hängen von den gewünschten Analysemethoden ab und können daher an dieser Stelle nicht spezifiziert werden. Als Ausgangsbasis kann die in Abb. 5.35 dargestellte Klasse (SISA_ModelAnalysis) dienen, die neben dem Simulationslauf-Namen eine Liste von Datensatzänderungen enthält. Die Änderungen können über die Angabe

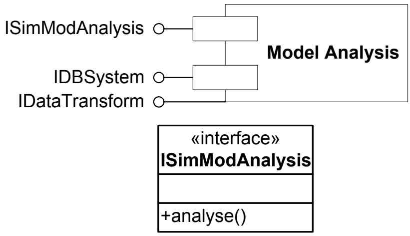


Abbildung 5.34: Schnittstellen zur Modellanalyse. Neben den Schnittstellen, die für einen transparenten Datenzugriff notwendig sind, bietet die Komponente eine Modellanalyse-Schnittstelle zur Aufnahme der eigentlichen Analyse-Operationen. Die Spezifikation der Schnittstellen IRepository und IDataTransform finden sich in den in Abb. 5.26 (Seite 137) und 5.27 (Seite 138).

des Datensatzes und die minimale und maximale Abweichung der Werte des Datensatzes (Klasse SISA_DatasetChange) spezifiziert werden[24].

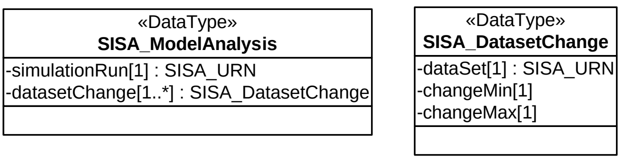


Abbildung 5.35: Einfaches Datenmodell zur Modellanalyse.

5.3 Gesamtarchitektur

Dieser Abschnitt liefert eine zusammenfassende Darstellung der entwickelten SISA-Architektur. Hierzu werden in Unterabschnitt 5.3.1 zunächst die statischen Aspekte der Architektur in Form eines Komponenten-Diagramms dargestellt und erklärt. Der anschließende Unterabschnitt (5.3.2, Seite 154) beschreibt die wichtigsten dynamischen Aspekte über Sequenzdiagramme.

[24] Auf die Angabe eines Datentyps für die Abweichungsangaben wird verzichtet, da hier sowohl Absolutwerte unterschiedlicher Typen als auch andere Datensätze vorgesehen werden könnten.

5.3.1 Komponenten

Die in diesem Kapitel entwickelte Gesamtarchitektur des SISA besteht aus insgesamt zwölf Komponenten – den Client nicht mitgezählt. Abbildung 5.36 zeigt die Komponenten und ihre gegenseitigen Abhängigkeiten als Komponentendiagramm. Die Schnittstellen und Verantwortlichkeiten jeder Komponente werden – wenn notwendig, ergänzt durch Anmerkungen über vorgeschlagene Datenstrukturen – im Folgenden zusammenfassend erklärt (ausführlichere Erläuterungen finden sich in den entsprechenden Unterabschnitten der Komponenten-Entwicklung ab Seite 96).

Documentation

Die Dokumentationskomponente (Documentation) ist verantwortlich für die Dokumentation und Verwaltung grundlegender *Hintergrundinformationen* über durchgeführte bzw. in der Durchführung befindliche Assessments. Zu den Hintergrundinformationen gehören Angaben über durchgeführte Projekte und Studien, beteiligte Personen und Organisationen, durchgeführte Simulationsläufe und verwendete Szenarien. Neben diesen Informationen stellt die Dokumentationskomponente auch ein Glossar bereit sowie einen Katalog, in dem Arbeitsschritte erklärt werden können, die häufig mit dem SISA durchgeführt werden (der so genannte 'Service Organizer Folder'). Ein Katalog, in dem die Nutzer des SISA nicht weiter spezifizierte Anmerkungen eintragen können, ist ebenfalls in dieser Komponente angesiedelt. Die Dokumentationskomponente bietet zwei Schnittstellen an: eine zur Verwaltung der internen Kataloge (IDocManager) und eine zur Abfrage der Kataloge nach bestimmten Kriterien (IDocDiscovery). Die Dokumentationskomponente enthält projektbezogenen Informationen über Ressourcen vom Katalogmanager und ist somit von dieser Komponente abhängig. Benutzt wird die Komponente lediglich vom Client.

Catalog Manager

Der Katalogmanager (Catalog Manager) ist für die Verwaltung und Bereitstellung von Metadaten über SISA-Ressourcen verantwortlich. Die Datenstruktur zur Speicherung von Metadaten orientiert sich an Standards: Grundlage für alle Ressourcenbeschreibungen ist der *Dublin Core Metadata Element Set (DCMES)* (ISO, 2003), dessen 15 Elemente mit relativ wenig Aufwand für jede Ressource ausgefüllt werden können. Detaillierte Beschreibungen zu Geodaten sollten sich am *ISO-Standard 19115* (ISO, 2000) orientieren, genauere Angaben zu Simulationsmodellen am *Content Standard for Computational Models* (ADEPT, 2001). Die Schnittstellen richten nach den *Catalog Services* des OpenGIS-Konsortiums und bieten Operationen zur Verwaltung und Abfrage von Katalogen (durch die Schnittstellen ICatManager und ICatDiscovery) sowie zur Abfrage von Zugriffsinformationen und zur Generierung eindeutiger Ressourcen-Namen (Schnittstelle ICatAccess). Der Katalogmanager wird sowohl vom Client benutzt als auch von der Dokumentations- und der Datenzugriffskomponente. Der Katalogmanager selbst ist unabhängig von anderen Komponenten.

Metadata Harvester

Der Metadaten-Sammler (Metadata Harvester) ist verantwortlich für die Durchsuchung eines Rechners (Hosts) nach Dateien mit Metadaten und die automatische Weitergabe der gefundenen Informationen an den Katalogmanager. Die Datenstruktur des Metadaten-Sammlers orientiert sich an der Spezifikation der Metadaten des Katalogmanagers. Über seine Schnittstelle (IMDHarvester) bietet der Metadaten-Sammler die Möglichkeit, die Sammlung von Metadaten zu starten und die gefundenen Informationen an den Katalogmanager weiterzuleiten. Um seine Aufgabe zu erfüllen, benötigt der Metadaten-Sammler Zugriff auf die Dateien des zu durchsuchenden Rechners. Für die Übermittlung der Metadaten greift er auf den Katalogmanager zu. Der Aufruf der Operationen kann durch den Client erfolgen oder in regelmäßigen Abständen durch einen entsprechenden Prozess.

Datasets

Das Paket der Datensätze (Datasets) ist eine lose, nicht direkt durch das SISA verwaltete Sammlung von Dateien und bietet keine über die SISA-Architektur definierten Schnittstellen an. Die Datenstrukturen sind ebenfalls nicht vorbestimmt. Daten, die für das SISA verwendet werden sollen und nicht im Datenbanksystem (s. u.) gespeichert werden, sollten aus Gründen der Interoperabilität etablierten Daten-Standards folgen. Zu jedem Datensatz sollte ein entsprechender Metadatensatz existieren. Bei der Verwendung neuer, für das SISA unbekannter Formate sollten der Datenzugriffskomponente entsprechende Operationen zum lesenden und schreibenden Zugriff auf Dateien dieses Formats hinzugefügt werden.

Data-Base System

Der Weg der direkten Datenintegration in das SISA geht über das Datenbanksystem (Database System). Das Datenbanksystem ist verantwortlich für die verwaltete Speicherung von Assessment-Daten (vgl. 'System-Daten', Unterkapitel 4.2.5, Seite 84). Zur Verwaltung der Daten bietet die Komponente in ihrer Schnittstelle (IDBSystem) Operationen zum Einfügen, Modifizieren, Abfragen und Löschen von Datensätzen. Die Komponente kann über ein eigenständiges Datenbank-Managementsystem oder einen Geodaten-Server realisiert werden.

Data Access

Die Datenzugriffskomponente (Data Access) ist für den lesenden und schreibenden Zugriff auf Daten verantwortlich sowie für die Transformation zwischen verschiedenen Datenformaten. Die Komponente erlaubt einen transparenten Zugriff sowohl auf die Daten im Datenbanksystem als auch auf die Daten, die in den Dateien der Datenbasis gespeichert sind. Um einen transparenten Datenzugriff zu gewährleisten, muss die Datenzugriffskomponente die gleiche Schnittstelle implementieren wie das Datenbanksystem (IDBSystem). Werden Daten in einem anderen als dem gespeicherten Format angefordert, übernimmt die Komponente automatisch die Umformatierung. Für die automatische Umwandlung sind entsprechende Transformationsfunktionen in der Komponente bereitzustellen. Die zur Transformation von Formaten verwendeten Funktionen sollten über eine gesonderte Schnittstelle (IDataTransform) auch expli-

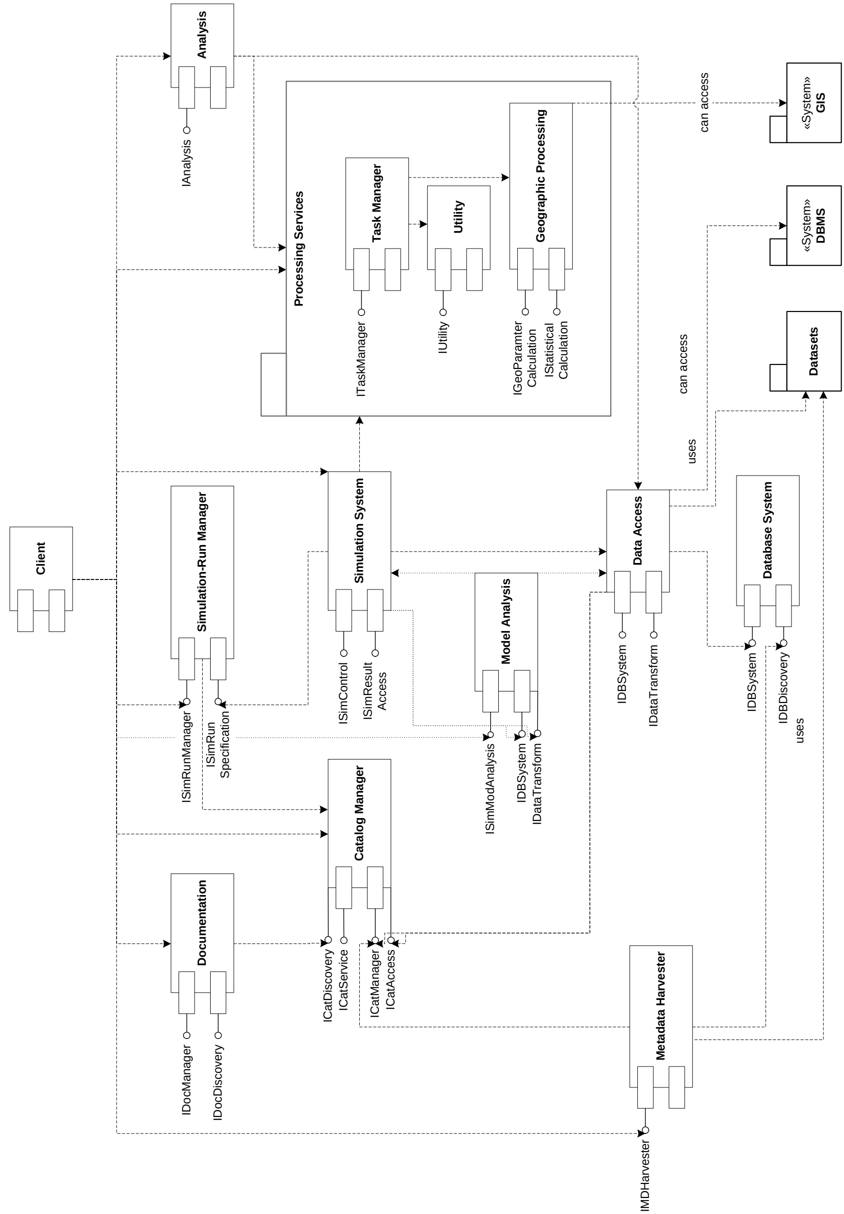


Abbildung 5.36: Komponenten der Architektur. Pfeile, die direkt auf Komponenten verweisen und nicht auf einzelne Schnittstellen, drücken die Abhängigkeit von allen Schnittstellen der entsprechenden Komponente aus. Erklärungen zu den Komponenten finden sich im Text.

zit aufrufbar sein (z. B. um Formate beim Datenaustausch unter Teilmodellen transferieren zu können). Der Zugriff auf Daten sollte stets über einen eindeutigen Bezeichner erfolgen; notwendige Zugriffsinformationen erhält die Datenzugriffskomponente über den Katalogmanager (genauer: über die Schnittstelle ICatAccess). Werden Daten geschrieben, sollten der Datenzugriffskomponente die entsprechenden Metadaten direkt mitgegeben werden, um eine konsistente Dokumentation der Datensätze zu gewährleisten. Beim Zugriff auf Daten aus dem Paket der Datensätze benötigt die Datenzugriffskomponente entsprechende Rechte, außerdem ist die Komponente zur Erfüllung ihrer Aufgaben vom Datenbanksystem abhängig. Alle Komponenten, die Zugriff auf die Assessment-Daten benötigen, sollten zur Sicherstellung der Datenintegrität und Nachvollziehbarkeit die Schnittstellen der Datenzugriffskomponente verwenden.

Simulation-Run Manager

Der Simulationslaufmanager (Simulation Run Manager) ist verantwortlich für die Verwaltung und Bereitstellung von Simulationslauf-Spezifikationen (Einstellungen, durch die sich ein Simulationslauf von einem anderen unterscheidet). Der Simulationslaufmanager bietet zwei Schnittstellen an: die erste (ISimRunManager) dient der Verwaltung von Simulationslauf-Spezifikationen, die zweite (ISimRunSpecification) der Abfrage einzelner Einstellungen und der Überprüfung, ob alle für ein Modell benötigten Spezifikationen vorhanden sind. Da Simulationsläufe (abstrakte) SISA-Ressourcen darstellen und daher durch einen eindeutigen Bezeichner zu repräsentieren sind, ist der Simulationslaufmanager zur Erfüllung seiner Aufgaben vom Katalogmanager (der für die Vergabe der Bezeichner zuständig ist) abhängig. Darüber hinaus gehören zu den Einstellungen für Simulationsmodelle, die ebenfalls über ihre eindeutigen Bezeichner zu referenzieren sind, auch Verweise auf Datensätze, deren Auswahl ebenfalls mit Hilfe der Dienste des Katalogmanagers erleichtert werden kann.

Simulationssystem

Die Simulationssystem-Komponente (Simulation System) ist verantwortlich für die Berechnung, Speicherung und Weitergabe von Simulationsergebnissen und stellt hierzu zwei Schnittstellen zur Verfügung: eine Schnittstelle (ISimControl), über die das Simulationssystem initialisiert wird und Simulationsläufe gestartet werden können und eine weitere Schnittstelle (ISimResultAccess), über die Simulationsergebnisse direkt abgefragt werden können. Die Möglichkeit auf – auch bereits gespeicherte – Simulationsergebnisse direkt über die Simulationssystem-Komponente zugreifen zu können, erhöht die Interoperabilität und Wiederverwendbarkeit der Modelle und Daten. Zur Erfüllung ihrer Aufgaben ist die Komponente von der Datenzugriffskomponente abhängig. Bei der Erstellung neuer Simulationsergebnisse benutzt die Komponente darüber hinaus die Dienste des Simulationslaufmanagers und Dienste aus dem Paket der Verarbeitungsdienste. Im Zuge von Modellanalysen wird der Datenzugriff über die Modellanalyse-Komponente abgewickelt.

Model Analysis

Die Komponente zur Modellanalyse (Model Analysis) ist verantwortlich für die Unterstützung bei der Analyse des Simulationssystems, insbesondere bei

der Durchführung von Sensitivitäts- und Unsicherheitsanalysen. Beim Start einer Modellanalyse (über die Schnittstelle ISimModAnalysis) übernimmt die Modellanalyse-Komponente die Kontrolle über die Durchführung von Simulationsläufen. Außerdem schaltet sie sich zwischen das Simulationssystem und die Datenzugriffskomponente, so dass alle Modelleingabedaten und Modellausgabedaten vor der Weitergabe kontrolliert und gezielt verändert werden können. Um den Datenzugriff für das Simulationsmodell transparent zu halten, muss die Modellanalyse-Komponente über ihre eigene Schnittstelle hinaus die gleichen Schnittstellen implementieren wie die Datenzugriffskomponente (IDBSystem und IDataTransform).

Analysis

Die Analyse-Komponente unterstützt den Modellbetreiber und Modellentwickler bei der Analyse von Simulationsergebnissen. Zu den Operationen, die diese Komponente (über die Schnittstelle IAnalysis) anbietet, sollte auf jeden Fall eine zur Visualisierung von Datensätzen bereitgestellt werden. Für weitergehende Analysen kann die Komponente auf das Paket der Verarbeitungsdienste zurückgreifen. Abhängig ist die Analyse-Komponente von der Datenzugriffskomponente, die die zu analysierenden Daten bereitstellt, und vom Katalogmanager, der umfassende Ressourceninformationen liefert (z. B. über die Formate der Datensätze, die analysiert werden). In der Analysekomponente können die Operationen implementiert werden, die für ein konkretes SISA gewünscht sind – also Dienste, die im Sinne der OGC-Dienstearchitektur (ISO/DIS 19119) zu den ‘user processing services’ zählen.

Processing Services

Die im Paket der Verarbeitungsdienste (Processing Services) zusammengefassten Komponenten bieten allgemeine Verarbeitungsdienste, die von den anderen Komponenten, unabhängig von einem speziellen SISA, verwendet werden können. Das Paket beinhaltet die Komponente zur Aufgabensteuerung (Task Manager) sowie die Komponenten zur (Geo-) Datenverarbeitung (Geographic Processing u. Utility). Die Komponente der Geodatenverarbeitung ist verantwortlich für die Bereitstellung von Diensten zur Verarbeitung geographischer Daten sowie den transparenten Zugriff auf die Funktionen eigenständiger Geo-Informationssysteme. Die von ihr angebotenen Schnittstellen (IGeoParameterCalculation und IStatisticalCalculation) sollten zumindest Operationen zur Transformation, Kombination und statistischen Auswertung von Rasterdaten bereitstellen. Weitere im Rahmen eines SISA nützliche Dienste, die Funktionen ohne georäumlichen Bezug anbieten, werden in der Komponente zur allgemeinen Datenverarbeitung (Utility) angeboten. Die Dienste dieser Komponente werden nicht weiter spezifiziert. Die Komponente der Aufgabensteuerung ist verantwortlich für die Bereitstellung von Diensten zum programmgesteuerten Aufruf anderer Dienste. Sie sollte über ihre Schnittstelle (ITaskManager) Operationen zur Definition und zum Ausführen von Diensten und Diensteketten bereitstellen.

Geo Processing

Utility

Task Manager

Wieder-verwendung

Die Dienste der Komponenten zur Geodatenverarbeitung und zur allgemeinen Datenverarbeitung können als ‘shared processing services’ im Sinne der OGC-Dienstearchitektur (ISO/DIS 19119) aufgefasst werden, also als Dienste, die nicht nur für ein spezielles SISA, sondern bereichsüberschreitend (beispielsweise für andere SISA oder zur Erstellung von Hilfsprogrammen) eingesetzt werden können.

5.3.2 Interaktionen

Die Interaktionen der Komponenten werden im Folgenden über Sequenzdiagramme verdeutlicht. Die als ‘Client’ bezeichneten Objekte in den Sequenzdiagrammen können sowohl menschliche Nutzer (Modellbetreiber, Modellentwickler etc.) als auch andere Software-Systeme (z. B. ein den Metadaten-Sammler anstoßender Hintergrundprozess) sein.

5.3.2.1 Simulationslauf

Das Sequenzdiagramm in Abb. 5.37 zeigt die Interaktionen der Komponenten bei der Erzeugung neuer Simulationsergebnisse (Testszenario /T60/).

init

Der Client startet die Initialisierung des Simulationssystems (Simulation System) für einen – zuvor durch eine Simulationslauf-Spezifikation definierten – Simulationslauf unter Angabe des eindeutigen Simulationslauf-Bezeichners über die Operation *init()*.[25] Während der Initialisierungsphase werden grundlegende Einstellungen des Systems vorgenommen. Die von den Simulationsmodellen benötigten simulationslaufspezifischen Einstellungen bezieht das Simulationssystem über die Operation *getSetting()* des Simulationslaufmanagers.

run

Nachdem die Initialisierung abgeschlossen ist, startet der Client den Simulationslauf über die Operation *run()*. Die im Laufe der Simulation benötigten Datensätze erhält das Simulationssystem von der Datenzugriffskomponente (Data Access). Hierzu nutzt das Simulationssystem die Operation *retrieveDataset()*. Der angeforderte Datensatz sollte über einen eindeutigen Bezeichner referenziert werden.

retrieve

access info

Die Datenzugriffskomponente selbst benötigt für den Zugriff auf den Datensatz zusätzliche Informationen über den angeforderten Datensatz (z. B. einen Datenbank-Namen oder das Format und den Speicherort einer Datei). Diese Informationen erhält die Datenzugriffskomponente – unter Verwendung des eindeutigen Datensatz-Bezeichners – über die Operation *getAccessInfo()* des Katalogmanagers (Catalog Manager).

database system

Die Datenzugriffskomponente bezieht den angeforderten Datensatz durch den Aufruf der Operation *retrieveDataset()* des Datenbanksystems (Database System) und gibt ihn dann direkt weiter an das Simulationssystem.

[25] Ein noch nicht initialisiertes Simulationssystem kann beim Eintreffen der Nachricht *run* die Initialisierung auch selbständig vornehmen.

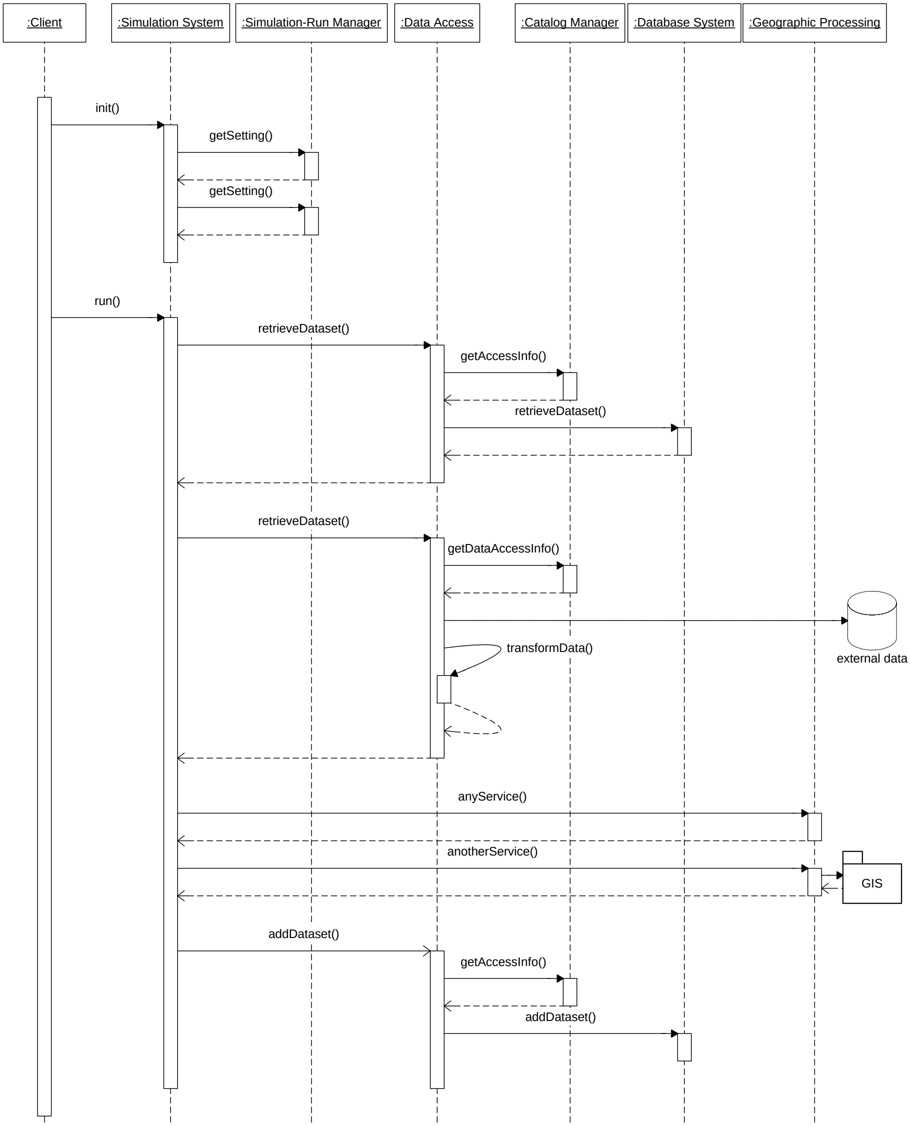


Abbildung 5.37: Architektur-Dynamik bei der Erzeugung von Simulationsergebnissen. Erklärungen finden sich im Text.

datasets Der zweite Aufruf von *retrieveDataset()* zeigt eine Aufruf-Sequenz für den Fall, dass die Daten weder im angeforderten Format noch im Datenbanksystem gespeichert sind. Nachdem die Datenzugriffskomponente durch *getDataAccess-Info()* festgestellt hat, dass die Daten nicht im Datenbanksystem zur Verfügung stehen, greift die Datenzugriffskomponente jetzt *direkt* auf den spezifizierten Datensatz – der sich im Paket *Datasets* befindet – zu. Danach ruft die Datenzugriffskomponente die interne Operation *transformData()* auf, um den Datensatz in das angeforderte Format umzuwandeln. Der Datensatz wird dem Simulationssystem dann im angeforderten Format übergeben.

trans-form

services Benötigt das Simulationssystem im SISA verfügbare Dienste für die (Geo-) Datenverarbeitung, so ruft es die entsprechenden Dienste auf. Dieser Vorgang wird durch den Aufruf der fiktiven Operation *anyService()* der Komponente *Geographic Processing* verdeutlicht. Der Aufruf der (ebenfalls fiktiven) Operation *anotherService()* verdeutlicht die Transparenz beim Aufruf einer GIS-Funktion: das Simulationssystem ruft die Operation der Komponente auf, die diesen Aufruf allerdings zur Bearbeitung an ein eigenständiges GIS weiterleitet und das Ergebnis anschließend zurückliefert – von der Weiterleitung erfährt der Aufrufer (in diesem Fall das Simulationssystem) nichts.

add dataset Die Speicherung von Simulationsergebnissen erfolgt analog zum oben beschriebenen lesenden Zugriff: das Simulationssystem ruft die entsprechende Zugriffs-Operation, also *addDataset()* auf. Daraufhin werden die für den Zugriff notwendigen Informationen über *getAccessInfo()* bezogen und der Datensatz (in diesem Fall) im Datenbanksystem ebenfalls über *addDataset()* gespeichert.

5.3.2.2 Metadaten-Sammlung

Die Integration von Metadaten in das System (Testszenario /T70/) geschieht u. a. über den Metadaten-Sammler. Die Interaktionen der Komponenten, die an der Sammlung von Metadaten beteiligt sind, werden über das Sequenzdiagramm in Abb. 5.38 verdeutlicht.

Bsp.: 2 Hosts Die Daten, die für ein Assessment von Interesse sind, können sich auf unterschiedlichen Rechnern (Hosts) befinden. Um die Daten dem System bekannt zu geben, ohne dass der Nutzer des Hosts explizit entsprechende Einträge im Katalogmanager vornehmen muss, werden Metadaten-Sammler (Harvester) eingesetzt. Auf jedem Rechner, der Daten für das SISA bereitstellt, sollte daher ein Metadaten-Sammler installiert sein[26]. Im Sequenzdiagramm in Abb. 5.38 sind zwei Hosts mit jeweils einem Harvester zu sehen (H1:Harvester und H2:Harvester).

harvest Wie in Beispiel (a) der Abb. 5.38 zu sehen ist, wird die Sammlung von Metadaten durch den Aufruf der Operation *harvest()* eines Metadaten-Sammlers

[26] Es ist auch möglich Metadaten über Rechnergrenzen hinweg zu sammeln; dies erfordert allerdings entsprechende Zugriffsrechte auf lokale Speichermedien, die nicht immer gewährt werden können.

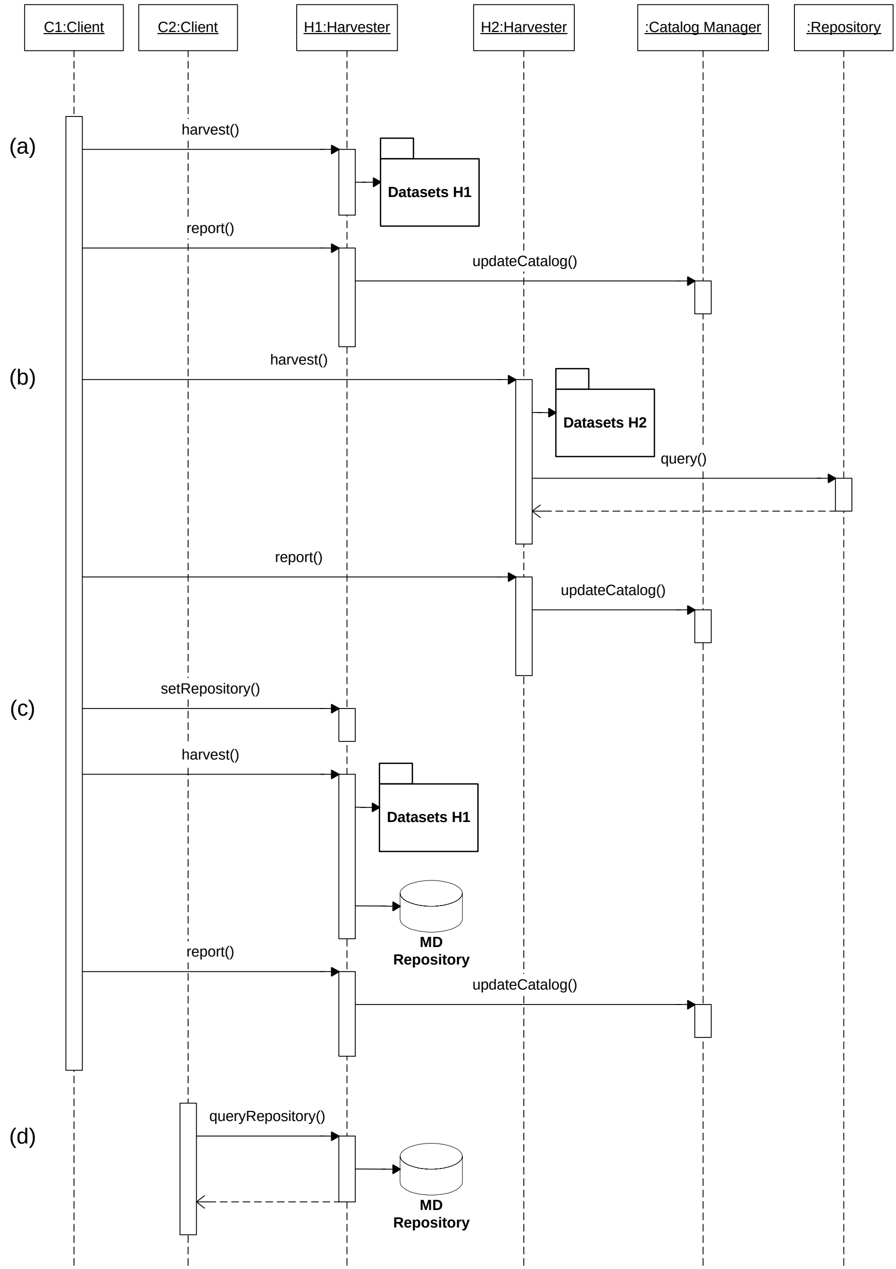


Abbildung 5.38: Architektur-Dynamik bei der Sammlung von Metadaten. Erklärungen finden sich im Text.

(Harvester) gestartet. Der Metadaten-Sammler durchsucht darauf hin die lokalen Dateien nach Metadaten und sammelt die gefundenen Metadaten intern.

report

Durch den Aufruf der Operation *report()* werden die gefundenen Metadaten dem Katalogmanager übermittelt. Hierzu verwendet der Metadaten-Sammler die Operation *updateCatalog()* des Katalogmanagers.

harvest

Die beschriebene Aufruf-Sequenz aus *harvest()* und *report()* wird für jeden Host durchlaufen, der Daten speichert, die für das SISA von Interesse sind.

Datenbanksystem-Suche

Der Metadaten-Sammler auf dem zweiten Host (H2:Harvester) durchsucht nicht nur die lokalen Dateien nach Metadaten: über die Operation *query()* des Datenbanksystems (Database System) sucht und sammelt er auch Metadaten, die sich evtl. im Datenbanksystem befinden. Beispiel (b) verdeutlicht die entsprechenden Aufruf-Sequenzen.

internes Repository

Sofern die Metadaten für die Daten eines Hosts auch auf dem Host selber gesammelt zur Verfügung stehen sollen, ist dies dem Harvester vor der Sammlung mitzuteilen. Wie in Beispiel (c) zu sehen ist, geschieht diese Mitteilung über die Operation *setRepository()*. Nach der Sammlung der Metadaten schreibt der Sammler ein eigenes (lokales) Repository. Auf ein lokales Repository können dann auch andere Clients zugreifen (z. B. der Nutzer des Rechners, auf dem das Repository gespeichert ist). Beispiel (d) zeigt den Aufruf der hierzu notwendigen Operation *queryRepository()* durch einen zweiten Client (C2:Client).

Aktualisierung

Der Aufruf zum Starten der Metadaten-Sammlung kann sowohl durch einen menschlichen Nutzer als auch durch eine andere Software erfolgen. Um die Metadaten aktuell zu halten, ist es ratsam, die Sammlung regelmäßig durchzuführen. Dies kann z. B. durch die Aufnahme des Aufrufs in einen regelmäßig ablaufenden Betriebssystem-Prozess sichergestellt werden.

5.3.2.3 Modellanalyse

Abbildung 5.39 zeigt das Sequenzdiagramm für einen Analyselauf. Im Fall der Modellanalyse schaltet sich die Modellanalyse-Komponente (Model Analysis) zwischen das Simulationssystem (Simulation System) und die Datenzugriffskomponente (Data Access) und kann damit die lesenden und schreibenden Datenzugriffe des Simulationssystems kontrollieren und manipulieren.

analyse

Eine Modellanalyse wird über die Operation *analyse()* der Modellanalyse-Komponente gestartet, die daraufhin die Kontrolle für Simulationsläufe übernimmt und zunächst das Simulationssystem über *init()* initialisiert. Nach der Initialisierung startet die Modellanalyse-Komponente den (ersten) Simulationslauf.

Transparenz

Alle Datenzugriffe des Simulationssystems werden nun über die Modellanalyse-Komponente abgewickelt. Diese Komponente bietet die gleichen Schnittstellen zum Datenzugriff an wie die Datenzugriffskomponente, der Zugriff auf die Daten bleibt daher für das Simulationssystem transparent und im Modell selbst müssen keine Änderungen für den Analyselauf vorgenommen werden.

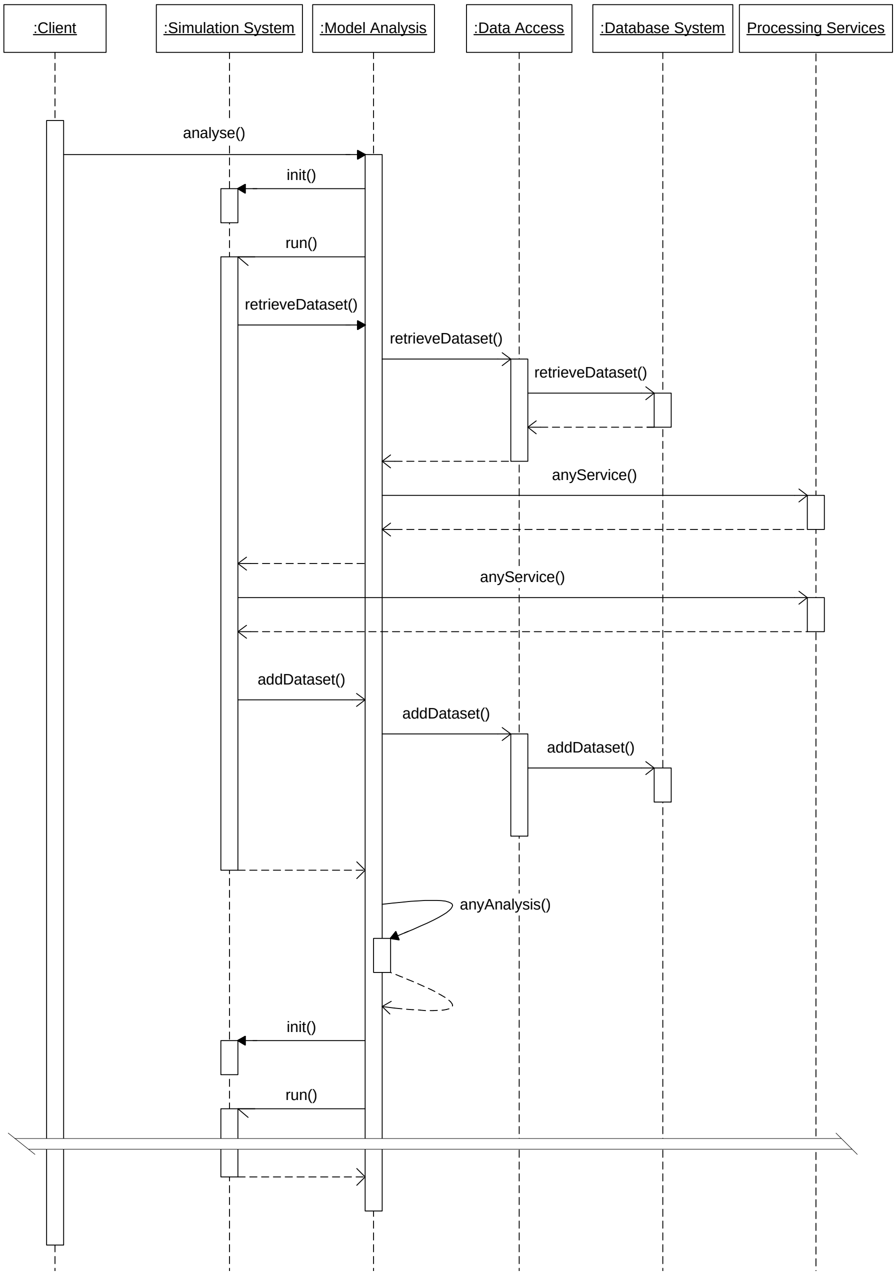


Abbildung 5.39: Architektur-Dynamik bei der Analyse eines Simulationssystems. Erklärungen finden sich im Text.

Das Lesen und Schreiben von Datensätzen erfolgt – wie bereits zu Abb. 5.37 (Seite 155) erklärt – über die Operationen *retrieveDataset()* und *addDataset()*.

services

Zur evtl. notwendigen Analyse oder gezielten Veränderung von Eingabedaten kann die Modellanalyse-Komponente, ebenso wie das Simulationssystem, die allgemein verfügbaren Dienste (Processing Services) nutzen.

analysis

Nach einem Simulationslauf werden die Simulationsergebnisse ausgewertet, was durch den Aufruf der Operation *anyAnalysis()* angedeutet wird.

Für Sensitivitäts- und Unsicherheitsanalysen müssen eine Vielzahl von Simulationsläufen durchgeführt werden. Bei jedem Simulationslauf werden die Eingabedaten durch die Modellanalyse-Komponente gezielt und nach einem bestimmten Verfahren geändert und danach die Ausgabedaten analysiert. Die Sequenz 'Initialisierung–Simulationsstart–Eingabedatenveränderung–Ausgabedatenanalyse' wird so lange wiederholt, bis ein definiertes Abbruchkriterium (z. B. die Anzahl durchgeführter Simulationen) erfüllt ist.

5.4 Fazit

Ziel

Ziel dieses Kapitels war die Entwicklung einer Software-Architektur für ein System zum integrierten simulationsbasierten Assessment (SISA), die als Ausgangsbasis für die Realisierung neuer Systeme herangezogen werden kann.

Ausgangsbasis

Den Rahmen für die Architektur-Entwicklung lieferten die in Kapitel 4 (Seite 69) definierten allgemeinen Anforderungen an ein SISA. Grundlage für die Abgrenzung der Architektur-Komponenten waren die in Kapitel 3 (Seite 25) identifizierten Komponenten existierender Systeme sowie die Dienste-Architektur des OpenGIS-Konsortiums (ISO/DIS 19119).

Ergebnisse

Zur Erfüllung der definierten Systemziele teilt die entwickelte Architektur das Gesamtsystem eines SISA in insgesamt zwölf Komponenten.

Die in der Systemdefinition geforderte Verwaltung verschiedener Betriebsmittel (Ressourcen) liegt in der Verantwortung des so genannten *Katalogmanagers*, der Metadaten zu allen SISA-Ressourcen (vgl. Abb. 4.2, Seite 76) bereithält. Für die Bereitstellung weiterer Hintergrundinformationen, z. B. über Projekte oder an einem Projekt beteiligte Personen, ist die *Dokumentationskomponente* verantwortlich. Die Verantwortlichkeit für die Berechnung neuer Simulationsergebnisse wird der *Simulationssystem-Komponente* übertragen. Diese Komponente konzentriert sich auf die Umsetzung des konzeptionellen Modells in ausführbaren Programmcode und nutzt zur Erfüllung ihrer Aufgabe die Dienste weiterer Komponenten, insbesondere die Dienste der *Datenzugriffskomponente* (um auf Datensätze zuzugreifen) und die des Simulationslaufmanagers (um die für einen Simulationslauf notwendigen Einstellungen zu erfragen). Häufig von Simulationsmodellen benötigte Funktionen zur *Geodatenverarbeitung* sowie zur *allgemeinen Datenverarbeitung* werden über separate Komponenten angeboten.

Die Einbettung des SISA in eine Software-Umgebung wird einerseits über den Austausch persistenter Daten erreicht, für die die *Datenzugriffskomponente* zuständig ist, und andererseits über die Möglichkeit der direkten Kopplung einzelner SISA-Komponenten mit externen Programmen über Programmierschnittstellen. Die Komponente zur *Geodatenverarbeitung* ist beispielsweise verantwortlich für die Kopplung des SISA mit eigenständigen Geo-Informationssystemen (GIS), die *Datenzugriffskomponente* für die Anbindung an externe Datenbanksysteme. Die in der Systemdefinition geforderte Interoperabilität mit solchen 'externen' Systemen (nicht-funktionale Anforderung /NF10/) wird damit explizit berücksichtigt. Die Austauschbarkeit von Systemteilen (/NF40/) sowie die Modifizierbarkeit (/NF30/) und Analysierbarkeit (/NF20/) des Systems wird insbesondere durch die klare Trennung der definierten Verantwortlichkeit der einzelnen Komponenten unterstützt. Der hohen Anforderung an das Qualitätsmerkmal der Austauschbarkeit wird insbesondere durch die Schnittstellendefinition der *Simulationssystemkomponente* Rechnung getragen. Die ebenfalls als wichtiges Qualitätsmerkmal eines SISA geforderte Transparenz, Nachvollziehbarkeit und Reproduzierbarkeit von Assessment-Ergebnissen (/NF50/) wird durch das Zusammenwirken der *Dokumentationskomponente*, des *Simulationslaufmanagers*, des *Katalogmanagers* und der *Metadaten-Sammler* sichergestellt.

Übersicht

Eine graphische Übersicht aller SISA-Komponenten sowie deren Verbindungen mit den SISA-Zielen und -Funktionen findet sich in den Abbildungen 5.1 (Seite 97) und 5.2 (Seite 99). Die Übersicht der spezifizierten Datenstrukturen und Schnittstellen ist im Anhang dokumentiert (Abbildung B.2, Seite 235 bzw. Abbildung B.3, Seite 236). Zum Abschluss des Kapitels folgt eine kurze Beschreibung der zwölf Komponenten der SISA-Architektur.

Simulationssystem

Die zentrale Komponente des *Simulationssystems* kapselt die Teilmodelle des Gesamtsystems und ist für die Berechnung neuer Simulationsergebnisse zuständig. Zur Erhöhung der Integrierbarkeit des Simulationsmodells bzw. dessen Teilmodelle, sollten diese in der Lage sein, zuvor berechnete Ergebnisse über eine entsprechende Schnittstelle zur Verfügung zu stellen.

Simulationslauf-Manager

Um die Nachvollziehbarkeit und Reproduzierbarkeit der Simulationsergebnisse zu gewährleisten, werden alle für einen Simulationslauf benötigten Einstellungen von einer separaten Komponente verwaltet: dem *Simulationslaufmanager*. Er versorgt das Simulationssystem auf Anfrage mit den notwendigen Informationen über Parametrisierungen, Optionen oder Eingangsdaten.

Der Ort der Speicherung von Eingangsdaten ändert sich in der Praxis von Zeit zu Zeit (z. B. im Zuge der Erweiterung oder des Austausches eines Daten-Servers). Um die Nachvollziehbarkeit und Reproduzierbarkeit von Ergebnissen dennoch zu gewährleisten, wird dem Simulationslaufmanager nicht der Ort (Pfad/Dateiname) der Speicherung eines Datensatzes angegeben, sondern ein *Ressourcen-Name*, der den Datensatz eindeutig identifiziert.

Daten-zugriff

Das Simulationssystem greift auf einen derart referenzierten Datensatz nicht direkt zu. Stattdessen wendet sich das Simulationssystem an die *Datenzugriffs-komponente*, die für die Bereitstellung von Datensätzen zuständig ist. Hierzu muss ihr lediglich der *Ressourcen-Name* des gewünschten Datensatzes sowie das *Format*, in dem der Datensatz geliefert werden soll, mitgeteilt werden. Diese Art des Datenzugriffs ermöglicht einen transparenten Datenzugriff sowie eine automatische Transformationen von Datenformaten. Das Format der Datenspeicherung wird somit von der 'internen' Repräsentation für das Simulationssystem getrennt. Dieser Zugriffsmechanismus erlaubt eine schrittweise Migration hin zu offenen Datenformaten. Die Schnittstelle zum Datenzugriff orientiert sich an den Spezifikationen der *OpenGIS Service Architecture (ISO/DIS 19119).*

Katalog-manager

Die Informationen, die die Datenzugriffskomponente zum Zugriff auf einen Datensatz benötigt (z. B. ein Pfad/Dateiname), werden vom *Katalogmanager*, der für die Verwaltung von *Metadaten* zuständig ist, bereitgestellt. Die Angabe der Zugriffsinformationen ist – außer für abstrakte Ressourcen – für jede Ressource zwingend erforderlich. Obligatorisch sind ebenfalls die Angabe eines eindeutigen Ressourcen-Namens und eines Kurztitels. All diese Informationen werden in die Ressourcen-Liste des SISA aufgenommen. Die Beschreibung jeder Ressource über die 15 Elemente des *Dublin Core Metadata Element Set* (ISO 15836) wird als Minimal-Anforderung angesehen. Weitergehende Beschreibungen zu Geodaten sollten sich nach dem Metadaten-Standard der ISO-19100-Reihe richten (ISO/DIS 19115), Metadaten zu Simulationsmodellen nach dem *Content Standard for Compulational Models*. Die Schnittstellendefinition sowie die Datenstruktur orientiert sich an den Spezifikationen der *OpenGIS Service Architecture (ISO/DIS 19119).*

Daten-basis

Die Datenzugriffskomponente greift auf die Daten innerhalb der Datenbasis zu. Die Datenbasis besteht aus einem *Datenbanksystem*, das eine verwaltete Datenspeicherung zulässt, und kann durch eine lose Sammlung von Dateien ergänzt werden. Die Integration der Dateien in das SISA erfolgt über Metadaten, die zu jeder Datei vorhanden sein sollten.

Meta-daten-Sammler

Die in der Datenbasis vorhandenen Metadaten, die nicht direkt vom Nutzer über den Katalogmanager eingegeben werden, werden von *Metadaten-Sammlern* eingesammelt und dem Katalogmanager übermittelt. Ein Metadaten-Sammler sollte auf jedem Rechner installiert sein, der Ressourcen für das SISA bereitstellt.

Doku-menta-tion

Projektbezogene Kurz-Informationen werden direkt im SISA vorgehalten, genauer: in der *Dokumentationskomponente*. In diesem 'Auskunftssystem' werden u. a. Daten über durchgeführte Simulationsläufe, beteiligte Personen und verwendete Szenarien hinterlegt und den Akteuren des SISA (Modellbetreiber, Modellentwickler, Entscheidungsträger, Interessenten) bereitgestellt.

Datenverarbeitung

Zur Steigerung der Wiederverwendbarkeit von Software sollten häufig wiederkehrende Funktionen nicht direkt in der Komponente des Simulationssystems implementiert, sondern in andere Komponenten ausgelagert werden. Zur Erhöhung der Interoperabilität und Austauschbarkeit sollte sich die Modularisierung von Funktionen an der Dienste-Taxonomie der *OpenGIS Service Architecture (ISO/DIS 19119)* orientieren. Im Rahmen des SISA lassen sich diesbezüglich drei relevante Bereiche (Komponenten) abgrenzen: Funktionen zur Bearbeitung geographischer Informationen, allgemeine Datenverarbeitungsfunktionen und Funktionen, die der Steuerung anderer Funktionen dienen. In der Architektur sind die Funktionen in den Komponenten *Geodatenverarbeitung*, *Datenverarbeitung* und *Aufgabensteuerung* lokalisiert. Die Komponente zur Geodatenverarbeitung sollte auch die Schnittstelle zu eigenständigen GIS darstellen.

Modell-Analyse

Zur Sensitivitäts- und Unsicherheitsanalyse von Simulationsmodellen ist eine gesonderte Komponente vorgesehen: die *Modellanalyse-Komponente*. Diese Komponente schaltet sich zur Modellanalyse als Schicht zwischen das Simulationssystem und die Datenzugriffskomponente. Auf diese Weise kann die Komponente die Eingabedaten für das Simulationsmodell gezielt verändern und die Ausgabedaten analysieren. Zur Modellanalyse müssen daher keine Änderungen innerhalb des Simulationssystems vorgenommen werden.

Ergebnisanalyse

Funktionen, die speziell für konkrete Assessments benötigt werden, werden der Komponente der *Ergebnisanalyse* zugeordnet.

Kapitel 6

Realisierung

Nachdem im vorigen Kapitel das Software-System eines SISA in seine wichtigsten Komponenten gegliedert wurde, beschäftigen sich die folgenden Abschnitte mit der Realisierung dieser abstrakten Komponenten, also mit deren *Implementierung* für ein konkretes System. Die prototypischen Implementierungen der Komponenten sollen die *Anwendbarkeit* der entwickelten Konzepte belegen. Unterschiedliche Realisierungsmöglichkeiten (z. B. über verschiedene Verteilungsplattformen) werden nicht gegenübergestellt. Ziel

Die Implementierungen der Komponenten erfolgten im Rahmen des GLASS-Modells; ein Modell zum Assessment der Auswirkungen des globalen Wandels auf die Nahrungsmittelproduktion und Wasserverfügbarkeit. Die beiden folgenden Abschnitte liefern zunächst eine kurze Beschreibung des Simulationssystems von GLASS sowie eine Übersicht zur Implementierung der SISA-Architektur, in die das GLASS-Simulationssystem eingebettet wird. Detaillierte Erklärungen zur Realisierung der einzelnen Architektur-Komponenten finden sich in Abschnitt 6.3 ab Seite 169. Das Fazit in Abschnitt 6.4 (Seite 199) fasst die wichtigsten Punkte der Komponenten-Realisierung zusammen. Übersicht

6.1 Beispielmodell GLASS

Das integrierte Modell *GLASS* (Global Assessment of Security; Alcamo u. a., 2001) ist ein Simulationssystem zur Quantifizierung der Beziehungen zwischen globalem Wandel und menschlicher Sicherheit unter Berücksichtigung natur- und sozialwissenschaftlicher Aspekte. Das politisch relevante Ziel ist die Darstellung umweltbezogener Bedrohungspotentiale für die menschliche Sicherheit und die Identifizierung von Strategien zur Risikominimierung. Die derzeitigen Prioritäten des GLASS-Modells liegen in der Verbindung extremer Klimaereignisse (z. B. Dürren) mit Risiken bei der Wasserversorgung und Nahrungsmit- Modell-Ziel

telproduktion. Die folgenden Unterabschnitte erklären kurz das Modellkonzept und die Rahmenbedingungen für die Realisierung des SISA.

Modellkonzept

Berechnungsprinzip

Die Klimavariabilität spielt bei der Analyse der Nahrungsmittelproduktion und Wasserverfügbarkeit eine wichtige Rolle. Aus diesem Grund werden über den *Klimavariabilitäts-Generator* (s. Abb. 6.1) szenarienbasierte Daten zu Temperatur und Niederschlag berechnet. Dieses Teilmodell berücksichtigt sowohl Daten zum historischen Klima (u. a. Angaben über Temperaturen und Niederschläge der letzten 100 Jahre auf einer monatlichen Basis) als auch Daten über mögliche Klimaänderungen, die von Klimamodellen für verschiedene Szenarien berechnet wurden. Die auf diese Weise generierten Klima-Daten werden anschließend vom Modell *WaterGAP* (Alcamo u. a., 2003a) zur Berechnung jährlicher Wasserverfügbarkeiten und vom Modell *GAEZ* (Fischer u. a., 2000) zur Berechnung möglicher Erträge wichtiger Feldfrüchte genutzt. Diese von Jahr zu Jahr schwankenden Indikator-Werte werden vom *Wasserstress-Modell* bzw. vom *Nahrungsmittelstress-Modell* zur Berechnung von 'Stresswerten' benutzt.[1] Bei der Berechnung des Nahrungsmittelstresses werden zusätzlich sozio-ökonomische Daten berücksichtigt (beispielsweise Angaben darüber, welche Feldfrucht in einer Region besonders wichtig ist oder wie hoch der Anteil an importierten Nahrungsmitteln in einer Region ist). Das *Sicherheitsmodell* verknüpft diese Stresswerte schließlich mit demographischen Angaben (z. B. der regionalen Altersstruktur und Einkommensverteilung), um Aussagen über die Wahrscheinlichkeit des Auftretens von Krisen und die potentiell von einer Krise betroffene Bevölkerung zu treffen.

Verwendung

Das GLASS-Konzept wird sowohl für Studien mit globaler als auch für Studien mit regionaler Abdeckung verwendet. Für eine erste globale Studie wurden 160 verschiedene Länder (Regionen) berücksichtigt (Alcamo u. a., 2001). Eine erste regionale Studie wurde für das Gebiet der russischen Föderation erstellt (R-GLASS; Alcamo u. a., 2003c) – in diesem Fall wurden 89 verschiedene Regionen unterschieden. Die Teilmodelle arbeiten auf unterschiedlicher räumlicher Auflösung. Geographisch explizite Berechnungen beruhen zumeist auf einer Auflösung von 0.5° * 0.5° geographischer Länge und Breite.

Rahmenbedingungen

Teilmodelle

GLASS verwendet sowohl Modelle, die sich bereits im Rahmen globaler Umweltforschung etabliert haben, als auch Modelle, die im Rahmen der GLASS-Entwicklung erstellt wurden. Die Modelle *WaterGAP*[2] (Alcamo u. a., 2003a;

[1] Die Stresswerte ergeben sich aus dem Verhältnis der jährlich schwankenden Wasserverfügbarkeiten bzw. Feldfruchterträge zu deren langjährigen Mittelwerten.

[2] Water – Global Assessment and Prognosis.

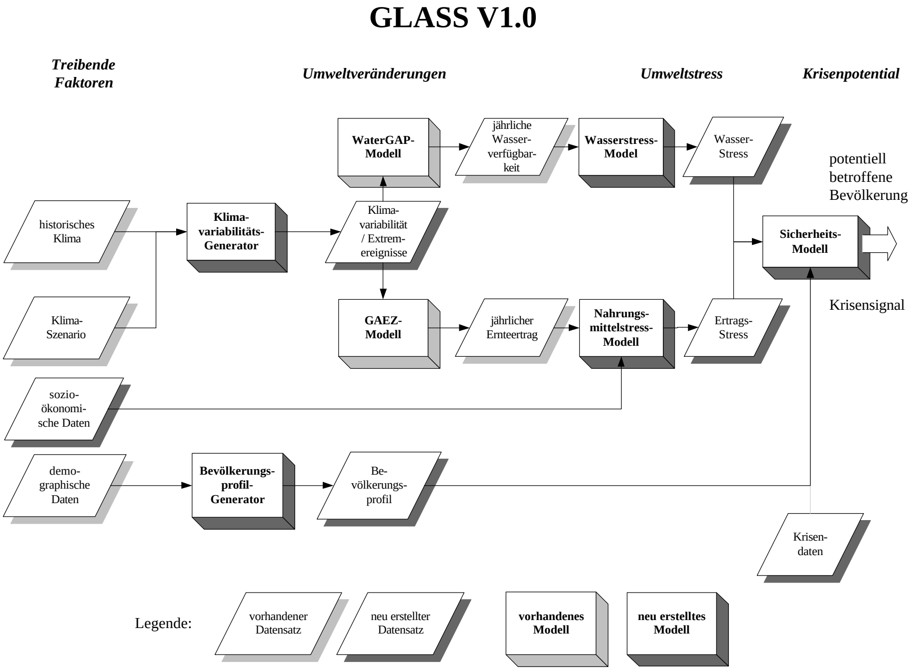


Abbildung 6.1: Struktur des GLASS-Modells. Ausgangsbasis für die Überprüfung der entwickelten SISA-Architektur ist ein loser Verbund von Teilmodellen, die zusammen das GLASS-Modell ausmachen. Die Teilmodelle wurden teilweise von anderen Projekten (Modell WaterGAP) bzw. Organisationen (Modell GAEZ) übernommen und teilweise neu erstellt. Die Kopplung erfolgt in diesem 'Gesamtmodell' lediglich über die von Teilmodellen gemeinsam genutzten Datensätze, beispielweise dem Datensatz zur jährlichen Wasserverfügbarkeit, der von WaterGAP berechnet und vom Wasserstress-Modell verwendet wird. Die Einstellungen, die von den einzelnen Modellteilen für die Simulation benötigt werden (z. B. Informationen über Datensätze) werden in dieser Form des Modells von jedem Modell separat verwaltet, was zu Inkonsistenzen führen kann (z. B. der Nutzung unterschiedlicher Versionen grundlegender Datensätze). Gemeinsam genutzte Datensätze müssen stets in dem Format vorliegen, das von den Modellen genutzt wird, so dass einige Datensätze (inhaltlich) redundant vorliegen müssen. Die Anwendung der Prinzipien der entwickelten SISA-Architektur sollen helfen, diese Nachteile zu überwinden.

Alcamo u. a., 2003b; Döll u. a., 2003) und *GAEZ*[3] (Fischer u. a., 2000) gehören zu den etablierten Modellen, während die anderen Modellteile im Rahmen der GLASS-Entwicklung entworfen wurden. *WaterGAP* wurde in C und C++ implementiert und *GAEZ* in FORTRAN, während die neu erstellten Teilmodelle in C++ realisiert wurden.

Datenformate

Aufgrund der Verwendung bereits existierender Simulationsmodelle müssen im GLASS-Modell unterschiedliche Datenformate berücksichtigt werden. Zu den Datenformaten gehören sowohl Textdateien im ASCII-Format als auch Geodatensätze im Binärformat des integrierten Modells IMAGE2 (Alcamo u. a., 1998b) in der Version2.2 (eingesetzt in WaterGAP) sowie Geodatensätze im ASCII-Austauschformat, im Raster-Format und im Polygon-Format des GIS ArcView[4]. Da GLASS rasterbasiert arbeitet, werden die letztgenannten ArcView-Daten im Polygon-Format nicht direkt vom Simulationsmodell verwendet. Sachdaten können aber als Tabelle behandelt werden. Sofern der geographische Bezug der Daten bei der Simulation explizit berücksichtigt werden soll (primärer Raumbezug), sind die polygonbezogenen Daten in rasterbasierte zu überführen. Die anderen angeführten Formate sind direkt bei der Konzeption der Implementierung der Datenzugriffskomponente zu berücksichtigen.

Verbrauchsverhalten

Bereits bei der prototypischen Implementierung der SISA-Architekturkonzepte sollte die Anforderung an das *Verbrauchsverhalten* berücksichtigt werden (s. Qualitätsziel-Bestimmung, Unterabschnitt 4.2.8, Seite 89). Einfache und effiziente Lösungen sind evtl. technisch 'schöneren' Lösungen vorzuziehen, um die Wiederverwendbarkeit der Software zu erhöhen. Bei der Betrachtung des Verbrauchsverhaltens spielt neben technologischen Gesichtspunkten auch die verwendete Software bzw. Middleware eine Rolle. Da das GLASS-Modell evtl. an Dritte weitergegeben werden soll (z. B. an die russischen Projektpartner der R-GLASS-Studie), sollte auf den Einsatz kostenintensiver Programme möglichst verzichtet und die Verwendung frei verfügbarer Software bevorzugt werden.

6.2 Komponenten-Übersicht

Im Rahmen des GLASS-Projektes wurden die zentralen Konzepte der SISA-Architektur getestet. Die Implementierung erfolgte prototypisch für ausgewählte Modellteile und Datensätze: Für einige *Teilmodelle* von GLASS (die Stressmodelle und das Sicherheitsmodell sowie WaterGAP) wurde die Schnittstelle implementiert, die für die Simulationssystem-Komponente definiert wurde (Operationen *init* und *run*). Auf diese Weise kann der Einfluss der Schnittstelle auf die Wiederverwendbarkeit, Austauschbarkeit und Interoperabilität verdeutlicht werden.

Teilmodelle

[3]Global Agro-Ecological Zones.

[4]Verwendet wurde die Version 3.2.

Das Gesamtmodell von GLASS, d.h. die Verbindung aller GLASS-Teilmodelle zu *einem* Software-System innerhalb der Simulationssystem-Komponente, wurde im Rahmen der Überprüfung der SISA-Architektur nicht realisiert. Die Implementierung der Konzepte, die sich auf den *Datenzugriff* beziehen, konzentrieren sich auf Geodaten des Modells WaterGAP. Datensätze

Die Realisierung der Konzepte und Komponenten erfolgte unter Nutzung verschiedener Programmiersprachen, Formate und Software-Systeme. Abbildung 6.3 (Seite 171) zeigt einige der Realisierungsaspekte in der Übersicht. Die Abbildung dient als Rahmen für die detaillierten Erklärungen der Realisierung des folgenden Abschnitts. Übersicht

6.3 Komponenten-Realisierung

6.3.1 Dokumentation

Die Dokumentationskomponente ist verantwortlich für die Bereitstellung grundlegender Hintergrundinformationen. Zu diesen, in Katalogform vorzuhaltenden Informationen, gehören Angaben über Personen/Organisationen, Projekte, Simulationsstudien, Simulationsläufe und Szenarien sowie ein Glossar, Beschreibungen zu typischen Arbeitsabläufen und kurze Anmerkungstexte (vgl. Unterabschnitt 4.2.5, Seite 84 u. Abschnitt 5.2.3, Seite 119). Daten

Die Daten der Dokumentationskomponente müssen dauerhaft und konsistent gespeichert werden und sollten einfach zu verwalten und abzufragen sein. Darüber hinaus muss die Dokumentationskomponente je nach Nutzer unterschiedliche 'Sichten' auf die Daten bereitstellen. Um allen Beteiligten an einem Assessment einen einfachen Zugang zu aktuellen Informationen zu bieten, sollten die Daten über ein Netzwerk (z. B. das Internet) abrufbar sein. Anforderung

Aufgrund der Anforderungen an die Datenhaltung und Datennutzung bietet sich die Verwendung eines *Datenbanksystems* an. Im Rahmen des GLASS-Projektes wurde das Datenbank-Managementsystem (DBMS) *MySQL* der schwedischen Firma *MySQL AB*[5] eingesetzt. MySQL ist ein Open-Source-Produkt im Sinne der GNU General Public License[6] und besitzt damit den Vorteil, dass keine Lizenzgebühren anfallen. Das fertige System kann somit auch ohne Probleme an andere Organisationen weitergegeben werden. Darüber hinaus ist MySQL – im Gegensatz zu sehr umfangreichen kommerziellen Systemen wie ORACLE oder IBM/DB2 – relativ einfach zu installieren und zu warten und bietet eine Vielzahl von Programmierschnittstellen (u. a. für die Sprachen C, C++, JAVA, Perl und PHP). Einige Leistungsmerkmale von DBMS werden von MySQL zwar (noch) nicht unterstützt (Transaktionen, Stored Procedures, DBS MySQL

[5]Startseite im Internet: `http://www.mysql.com`

[6]Startseite im Internet: `http://www.gnu.org/licenses`

Trigger, referenzielle Integrität), für den Einsatz in der Dokumentationskomponente des SISA kann auf diese aber verzichtet werden.[7]

Tabellen

MySQL ist ein relationales DBMS, d. h. die Daten werden in Tabellenform gespeichert.[8] Die in Kapitel 5 definierten SISA-Daten müssen daher in eine 'normalisierte' Tabellenform gebracht werden.[9] Abbildung 6.2 zeigt als Beispiel die Tabellen, über die die 'Anmerkungen' (engl. annotations) gespeichert werden (vgl. Abb. 5.19, Seite 128).

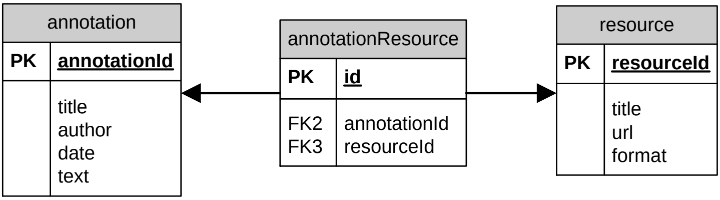


Abbildung 6.2: Tabelle zur Speicherung der Anmerkungen und deren Beziehung zu anderen Tabellen.

Beispiel

Die im Rahmen der Architekturentwicklung definierte Datenstruktur für Anmerkungen wird umgesetzt durch das Zusammenspiel von drei Tabellen: die erste Tabelle ('annotation') enthält Titel und Inhalt der Anmerkung sowie die Angaben zu Autor und Datum. Jede Anmerkung wird – genauso wie jede Ressource – darüber hinaus durch eine eindeutige Bezeichnung, den Primärschlüssel (engl. primary key, PK), gekennzeichnet. Die Verbindung von Anmerkungen zu Ressourcen wird über eine dritte Tabelle ('annotationResource') realisiert. Jede Zeile dieser Tabelle enthält den Primärschlüssel einer Anmerkung und einer Ressource (die Primärschlüssel werden hier als 'Fremdschlüssel' (engl. foreign key, FK) bezeichnet, da sie sich auf die Schlüssel anderer (fremder) Tabellen beziehen).

SQL

MySQL selbst besitzt keine graphische Benutzungsoberfläche. Die Definition und Abfrage von Tabellen und die Verwaltung von Datenbanken geschieht über ein Kommandozeilen-Werkzeug und die Anweisungen der *Structured Query Language (SQL)*. Da die Nutzung eines Datenbanksystems über Kommandozeilen wenig komfortabel ist, werden zusätzliche Werkzeuge zur Bedienung von MySQL angeboten. Ein Werkzeug zur Verwaltung von MySQL-Datenbanken ist beispielsweise *phpMyAdmin*[10], das auch im Rahmen der GLASS-Entwicklung

Werkzeuge

[7] Zu den Vor- u. Nachteilen von MySQL gegenüber anderen DBMS siehe z. B. Greenspan und Bulger (2001).

[8] Nähere Informationen zum Prinzip relationaler Datenbanksysteme und einen Vergleich mit objektorientierten Systemen findet sich in Balzert (1996).

[9] Zur Normalisierung s. z. B. Balzert (1996) oder Schwinn (1992).

[10] Startseite im Internet: `http://phpmyadmin.sourceforge.net`

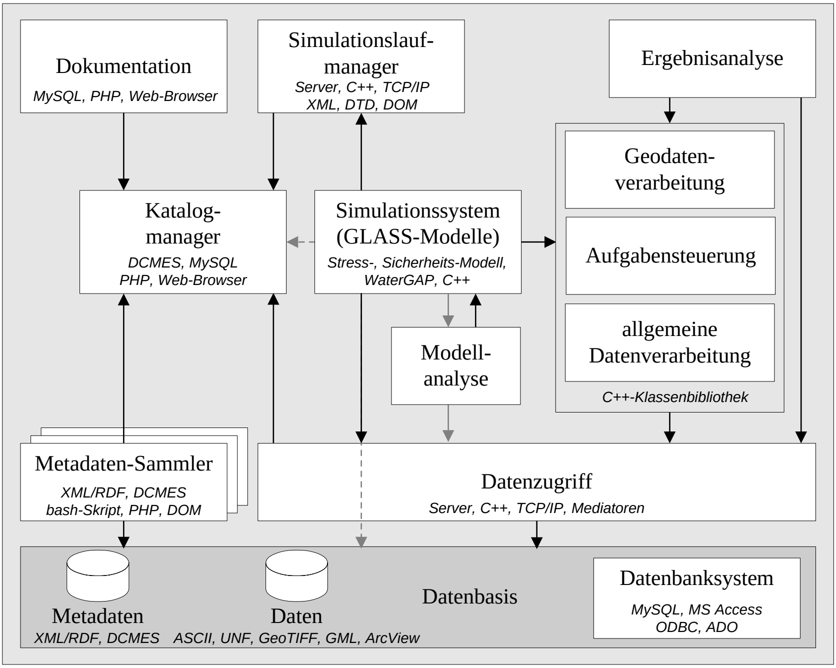


Abbildung 6.3: Übersicht zum Realisierungsbeispiel der SISA-Komponenten. Die für das *Simulationssystem* definierte Schnittstelle wird durch einige Teilmodelle von GLASS, z. B. vom Modell *WaterGAP*, unter Verwendung der Programmiersprache C++ realisiert. Die *Datenzugriffskomponente* ist ebenfalls in C++ implementiert und läuft als *Server* auf einem Host unter *Microsoft Windows 2000*. Die Anforderung zum transparenten Zugriff auf Datensätze geschieht über das *Internet-Protokoll (TCP/IP)*. Zur Datenspeicherung stehen sowohl Datenbanksysteme (*MySQL* und *MS Access*) als auch verschiedene proprietäre (UNF) und standardisierte (z. B. GeoTIFF) Dateiformate zur Verfügung. Der Katalogmanager nutzt zur Speicherung von Zugriffsinformationen und Metadaten (ebenso wie die Dokumentationskomponente) eine *MySQL*-Datenbank, die über einen *Web-Browser* bedient werden kann. Als Metadaten-Elementsatz werden die 15 Elemente des *Dublin Core Metadata Element Set (DCMES)* verwendet. Die Speicherung lokaler Metadaten erfolgt über die *Extensible Markup Language (XML)* in Verbindung mit dem *Resource Description Framework (RDF)*. XML wird ebenfalls zur Speicherung der Simulationslaufspezifikationen innerhalb des als Server implementierten *Simulationslaufmanagers* verwendet. Wiederverwendbare Datenverarbeitungsfunktionen wurden in Form von C++-Klassenbibliotheken implementiert.

eingesetzt wurde. Die Eingabe neuer Daten in das System ist aber auch mit diesem Werkzeug noch recht rudimentär.

PHP u. Browser

Für die Eingabe und Anzeige von Daten wurde daher ein anderer Weg gewählt: die Verbindung von MySQL mit der Programmiersprache *PHP*[11] und einem Web-Browser. Daten, die über ein Formular in einen Web-Browser (beim Client) eingegebenen werden, können über den Web-Server direkt an einen PHP-Interpreter weitergeleitet werden, der sie seinerseits in eine MySQL-Datenbank schreibt.

Voraus-setzung

Um diesen Weg der Datenverwaltung zu gehen, muss lediglich ein Web-Server (z. B. der frei verfügbare 'Apache'-Server[12]), MySQL und ein PHP-Interpreter (ebenfalls frei verfügbar[13]) auf dem Server installiert sein. Abbildung 6.4 verdeutlicht das Prinzip der Zusammenarbeit dieser Systeme noch einmal. Weitere Informationen hierzu finden sich z. B. bei Greenspan und Bulger (2001).

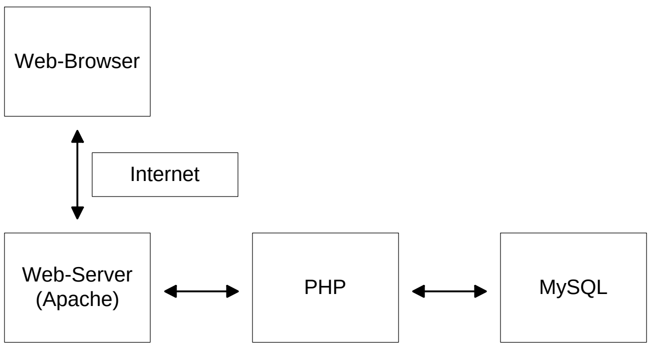


Abbildung 6.4: Datenbankzugriff über Web-Browser. Bei der gewählten Realisierung greift der Nutzer über den Web-Browser und das Internet auf den Web-Server zu. Der Web-Server (in diesem Falle 'Apache') sendet die Daten weiter an den PHP-Interpreter, der sie auswertet und über die API von MySQL lesend bzw. schreibend auf die Datenbank zugreift.

Erfas-sung u. Anzeige

Eine einfache Maske zur Erfassung von Personen/Organisationen ('responsible party', vgl. Abb. 5.14, Seite 123) über einen Web-Browser ist in Abbildung 6.5 zu sehen. Die Anzeige des Datenbank-Inhalts über den Web-Browser zeigt Abb. 6.6 (Seite 174).

Sichten

Der in Unterabschnitt 4.2.7 (Seite 89) aufgestellten Forderung zur Verwendung von Web-Technologien (/B10/) wird durch dieser Vorgehensweise Rech-

[11] PHP ist eine rekursive Abkürzung und steht für 'PHP: Hypertext Processor'.
[12] Startseite im Internet: `http://httpd.apache.org`
[13] Startseite im Internet: `http://www.php.net`

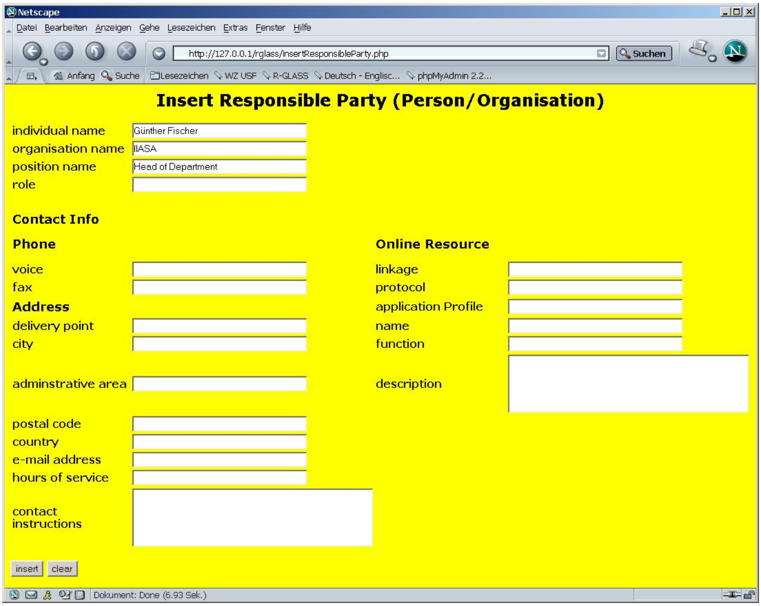


Abbildung 6.5: Dokumentation der Daten über Personen/Organisationen. Der Nutzer (Client) braucht im Web-Browser lediglich eine entsprechende (PHP-) Seite aufzurufen, um die Daten eingeben und an den Web-Server senden zu können. Betätigt der Nutzer die Schaltfläche zum Absenden des Formulars ('insert'), so wird dessen Inhalt automatisch zum Web-Server und über den PHP-Interpreter an die MySQL-Datenbank gesendet.

nung getragen. Die Bereitstellung unterschiedlicher 'Sichtweisen' auf die Daten des Systems (/B20/) kann über unterschiedliche Web-Seiten realisiert werden: Modellbetreibern und Entwicklern, die sich im lokalen Netz (Intranet) der Dokumentationskomponente befinden, können Seiten zur Verfügung gestellt werden, die *alle* Daten einer Tabelle anzeigen, Interessenten außerhalb des Intranet bekommen hingegen nur eine Auswahl aller verfügbaren Informationen über den Web-Server zugesendet. Darüber hinaus kann der Zugriff auf Web-Seiten relativ einfach durch Passwörter geschützt werden.

6.3.2 Katalogmanager

6.3.2.1 Ressourcen-Identifizierung

Der Katalogmanager ist verantwortlich für die Verwaltung und Bereitstellung von Metadaten über SISA-Ressourcen (s. Abschnitt 5.2.1 (Seite 96)). Zu den

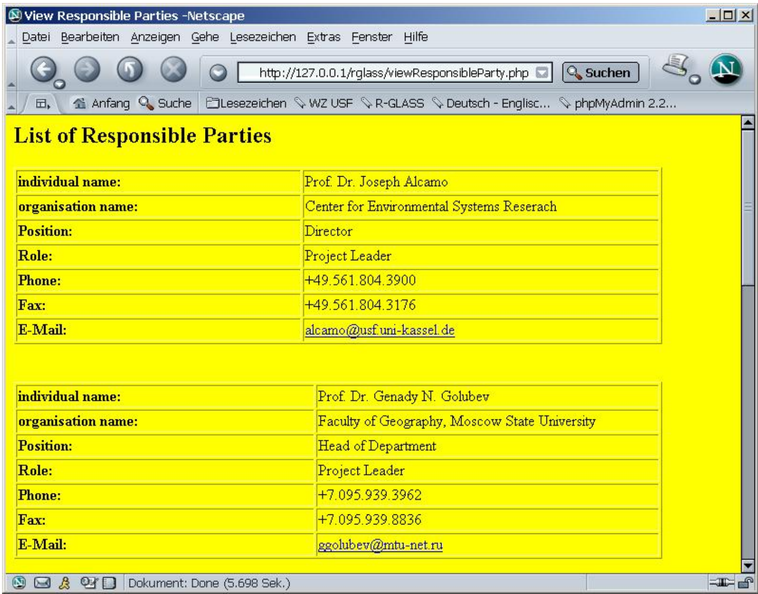


Abbildung 6.6: Anzeige der Daten über Personen/Organisationen. Ruft der Nutzer die Web-Seite zur Anzeige der Daten auf, so wird über den Web-Server und den PHP-Interpreter eine Datenbank-Anfrage ausgeführt. Die Ergebnisse werden dann automatisch an den Browser zurück gesendet und dort angezeigt. Ein Ausschnitt des Quelltextes dieser PHP-Seite ist im Anhang D.1 (Seite 245) dokumentiert.

Metadaten gehören ein eindeutiger Identifikator jeder Ressource sowie Informationen, die für den technischen Zugriff auf die Ressource benötigt werden.

IETF Für die Lokalisierung und Identifizierung von Ressourcen hat die Internet Engineering Task Force (IETF)[14] einige Vorschläge (RFCs[15]) erarbeitet: den *Uniform Resource Identifier* (RFC 2396, Berners-Lee u. a., 1998), den *Uniform Resource Locator* (RFC 1738, Berners-Lee u. a., 1994) und den *Uniform Resource Name* (RFC 2141, Moats, 1997).

URI Ein *Uniform Resource Identifier (URI)* ist definiert als „a compact string of characters for identifying an abstract or physical resource.“ (Berners-Lee u. a., 1998) Das ‘Einheitliche’ an diesen Identifikatoren ist die generische Syntax, mit der die identifizierenden Zeichenketten zusammengesetzt werden. Als ‘Ressource’ wird alles verstanden, was eine Identität hat; das können nicht nur elek-

[14]Startseite im Internet: `http://www.ietf.org`
[15]Requests for Comments. Siehe Grundlagenkapitel.

tronische Ressourcen wie beispielsweise Dateien sein, sondern auch Menschen, Dienstleistungen oder Firmen. Der Identifikator wird als Objekt betrachtet, das als Referenz auf etwas dienen kann, das eine Identität hat – im Falle des URI ist dieses Objekt eine Zeichenkette, die bestimmten Regeln zu folgen hat. URIs können[16] klassifiziert werden in solche zur *Lokalisierung* und solche zur *Bezeichnung* von Ressourcen (vgl. Abb. 6.7).

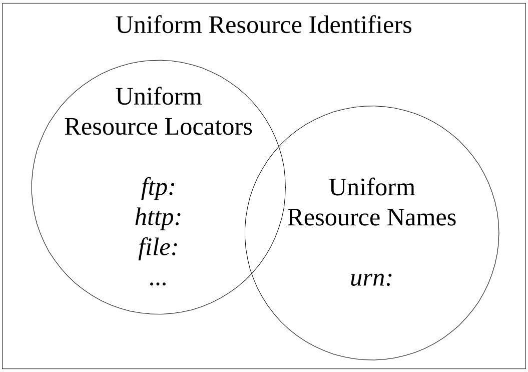


Abbildung 6.7: Uniform Resource Identifiers. Zu den Uniform Resource Identifiers (URIs) zählen sowohl die Uniform Resource Locators (URLs) als auch die Uniform Resource Names (URNs). Ziel aller Identifizierer ist die Identifizierung abstrakter und physikalischer Ressourcen in Form einer Zeichenkette. Zu den URLs gehören WWW-Adressen und Adressen von Datei-Servern (ftp-Servern). URNs werden u. a. zur Beschreibung von Buchtiteln benutzt (z. B. url:isbn:3-8266-0805-4).

Ein *Uniform Resource Locator (URL)* verweist (zeigt) auf eine Ressource eher durch den Mechanismus des *Zugriffs* auf diese Ressource, als durch andere Attribute wie dem Ressourcen-Namen. Eine WWW-Adresse ist ein Beispiel für einen URL. URLs sind sowohl Grundlage für das World Wide Web (WWW) als auch entscheidend für andere Internet-Technologien (z. B. XML). Der Aufbau eines URL folgt einer generischen Syntax der Form: URL

<scheme>‘:’<scheme-specific-part>

Der erste Teil (scheme) stellt das *Schema* der URL dar, dem, durch einen Doppelpunkt getrennt, eine *Zeichenkette* (scheme specific part) folgt und dessen Interpretation vom Schema abhängt (Berners-Lee u. a., 1994). Schemata können bei der *Internet Assigned Numbers Authority (IANA)*[17] registriert werden.

[16] Zu unterschiedlichen Sichtweisen und Interpretationen des URI-Konzepts s. Mealling und Denenberg (2002).

[17] Startseite im Internet: `http://http://www.iana.org`

Unter den derzeit etwa 40 registrierten Schemata sind, neben dem bereits erwähnten Schema ‘http’ (das für ‘Hyper-Text Transfer Protocol’ steht), z. B. ‘ftp’ (zum Austausch von Dateien), ‘telnet’ (für den entfernten Zugriff auf Rechner) und ’file’ (zur Angabe des Speicherortes von Dateien).[18] Bei der Registrierung der Schemata wird auch die genaue Syntax festgelegt, der die schemaspezifischen Zeichenketten folgen müssen.

URL-Zuweisung

Zur Steigerung der Nachvollziehbarkeit und Überprüfbarkeit von Assessment-Ergebnissen und zum einfachen Zugriff auf die Ressourcen über die Datenzugriffskomponente sollten allen digital zugreifbaren SISA-Ressourcen URLs zugewiesen werden. Der folgende URL kann beispielsweise die Datei mit den Länderinformationen für das das Simulationsmodell IMAGE2 referenzieren:

file://Usf-ws14/grid/data/general/modelInput/GCOUNTRY.UNF2

Die Länderinformationen sind demnach in Form einer Datei (Schema ‘file’) gespeichert und befinden sich auf dem Rechner Usf-ws14 unter dem angegebenen Pfad.

URN

Ein *Uniform Resource Name (URN)* stellt, im Gegensatz zu einem URL, keinen Verweis auf den Ort einer Ressource bereit, sondern einen (global) eindeutigen und beständigen (persistenten) Namen. Die genaue Definition des IETF lautet wie folgt: „Uniform Resource Names (URNs) are resource identifiers with the specific requirements for enabling location independent identification of a resource, as well as longevity of reference.“ (Daigle u. a., 2002)

URNs bestehen, wie die URLs, aus Zeichenketten und müssen einer vorgegebenen Syntax entsprechen (Moats, 1997):

‘urn:’<NID>‘:’<NSS>

‘NID’ ist der so genannte ‘Namespace Identifier’, der – vergleichbar mit dem ‘scheme’ beim URL – die *syntaktische* Interpretation des ‘Namespace Specific String’ (NSS) festlegt. Die Registrierung der Namensräume ist ebenfalls über die IANA möglich.[19] Unter den derzeit knapp 20 registrierten Namensräumen findet sich beispielsweise auch derjenige zur Angabe der internationalen Standardbuchnummer (ISBN). Die Angabe eines Buchtitels in Form eines URN sieht, obiger Syntax folgend, so aus: urn:isbn:3-8266-0805-4.

SISA Namensraum

Im Rahmen des integrierten Assessments werden viele Ressourcen zwischen unterschiedlichen Organisationen ausgetauscht: Daten, Modelle, Teilmodelle etc. Um die Wiederverwendbarkeit der Ressourcen zu erhöhen und Assessment-Ergebnisse besser nachvollziehbar und vergleichbar zu machen, wäre die Einführung eines definierten, einheitlichen und bei der IANA registrierten Namensraumes für diese Interessensgemeinschaft sicherlich erstrebenswert. Die

[18] Die offizielle Liste der Schemata wird unter `http://www.iana.org/assignments/uri-schemes` veröffentlicht.

[19] Die offizielle Liste der Namensräume wird unter `http://www.iana.org/assignments/urn-namespaces` veröffentlicht.

für eine formale Registrierung notwendige Koordination unterschiedlicher Interessensschwerpunkte ist allerdings sehr aufwendig. Ein formal bei der IANA registrierter Namensraum für SISA-Ressourcen ist daher (vorerst) nicht zu erwarten. Für solche Fälle wurden in den Spezifikationen der URN so genannte 'experimentelle' Namensräume eingeführt. Für die Bezeichnung dieser nicht bei der IANA registrierten Namensräume wird die folgende Form vorgeschrieben: x-name

'x-'<NID>

'NID' steht hier wieder für 'Namespace Identifier', also für die Bezeichnung des Namensraumes. Bezeichnungen dieser Art sind für 'interne oder eingeschränkte experimentelle Kontexte' gedacht, für die – im Gegensatz zu den formell registrierten URNs – keine Vorkehrungen zur 'Vermeidung von Kollisionen' mit anderen Namensräumen getroffen werden (Daigle u. a., 2002).

Für die Bezeichnung von SISA-Ressourcen am *wissenschaftlichen Zentrum für Umweltsystemforschung (WZ-USF)* wurde ein *experimenteller Namensraum* mit dem NID 'wzusf' eingeführt; URNs für SISA-Ressourcen beginnen daher immer mit der Zeichenfolge x-wzusf

'urn:x-wzusf:'

Zu einem URN gehört, neben der Bezeichnung des Namensraumes, auch die Definition der Syntax für die spezifische Zeichenkette (NSS). Die Definition dieser Syntax für 'x-wzusf' fußt auf der Analyse der unterschiedlichen SISA-Ressourcen aus Unterabschnitt 4.1.2 (Seite 72) und hat die folgende Form: Syntax

<resTyp>-<resUTyp>.<kTitel>-V<verNr>.<relNr>.<for>.<med>

Die Zeichenkette wird damit definiert über den Typ und den evtl. vorhandenen Untertyp der Ressource (resTyp, resUTyp), einen Kurztitel (kTitel), eine Versions- und Releasenummer (verNr, relNr) sowie das Format der Ressource (format) und das Medium, auf dem sie gespeichert ist (med). Als Ressourcen-Typen sind derzeit definiert: Datensatz (resTyp = 'ds'), Datensatz-Sammlung (dsc), Dokument (doc), Software (sw) sowie die abstrakten Ressourcen (ares). Zu den Untertypen gehören z. B. Simulationsmodell-Eingabedaten (ds-min) und -Ausgabedaten (ds-mout), Präsentationen (doc-prs) sowie Simulationsmodelle (sw-mod). Der URN Beispiel

urn:x-wzusf:ds-min.image22countryGrid-V1,0.unf.hd

identifiziert beispielsweise den bereits erwähnten Datensatz (ds) mit den Länderinformationen für IMAGE2.2 (Kurztitels 'image22countryGrid'), der primär als Eingabedatensatz für Simulationsmodelle (min) genutzt wird und in der Version '1.0' sowie im Format 'unf' an einem hier nicht spezifizierten Ort (vgl. URL-Bsp.) auf der Festplatte ('hd') gespeichert ist.

URN-Erzeugung

Allen SISA-Ressourcen sollte ein derartiger URN zugewiesen werden. Zur Generierung neuer URNs auf Basis der beschriebenen Syntax wurde im Rahmen des GLASS-Projektes die in Abb. 6.8 dargestellte Web-Seite entwickelt, die auch eine Übersicht über die derzeit definierten Ressourcen-Untertypen gibt. Um zu gewährleisten, dass der URN noch nicht vergeben ist, müsste, über die derzeitige Funktion der Web-Seite hinaus, ein Abgleich mit der Ressourcen-Liste des Katalogmanagers vorgenommen werden (vgl. die Schnittstellendefinition des Katalogmanagers in Abb. 5.6, Seite 107).

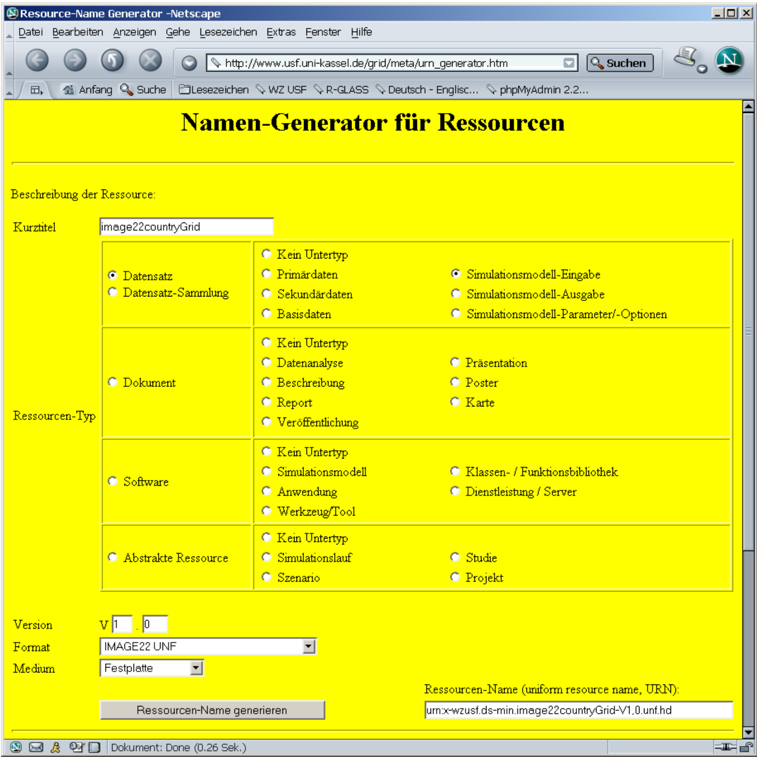


Abbildung 6.8: Web-Seite zur Generierung eindeutiger Namen. Die Bezeichnung von Ressourcen erfolgt über Uniform Resource Names (URNs). Zur Generierung eines solchen Namens ist jede Ressource mit einer Kurzbezeichnung zu versehen. Weiterhin müssen Angaben zum Typ der Ressource gemacht werden sowie zum verwendeten Speicherformat und Speichermedium. Wird die Schaltfläche 'Ressourcen-Name generieren' betätigt, erscheint im entsprechenden Feld der URN.

6.3.2.2 Ressourcen-Liste

Die im Projekt verwendeten Ressourcen können über eine Ressourcen-Liste zur Verfügung gestellt werden (/F60/). Zur Identifizierung der Ressourcen besitzt jeder Listeneintrag auf jeden Fall einen URN und – sofern es sich nicht um eine 'abstrakte' Ressource handelt – einen URL. Der URN dient der eindeutigen Bezeichnung und der URL der eindeutigen Lokalisierung der Ressource. URN u. URL

Der Katalogmanager ist auch für die Bereitstellung von Informationen zuständig, die die Datenzugriffskomponente für den Zugriff auf Ressourcen benötigt. Diese Zugriffsinformationen sollten sich bereits aus dem 'Format'-Teil des URN ergeben (in obigem Beispiel war das die Zeichenkette 'unf'). Werden über diese Angabe hinaus zusätzliche Informationen für den Zugriff benötigt, sollten sie ebenfalls in der Ressourcenliste zur Verfügung stehen. Im angeführten Beispiel der Daten mit der Länderkennung wurde das Format als 'unf'-Format angegeben. Dieses Format lässt sich aber weiter unterteilen, so dass die entsprechende Information mit in die Ressourcen-Liste aufgenommen wurde. Um die Ressourcen-Liste lesbarer zu gestalten und um einfacher nach Ressourcen suchen zu können wird noch ein weiteres Feld mit in die Liste aufgenommen: der 'Titel' (Name) der Ressource. Abbildung 6.9 zeigt die resultierenden Einträge, die für jede Ressource gemacht werden sollen, anhand eines Ausschnitts einer Ressourcen-Liste. Zugriffsinfo Titel

6.3.2.3 Metadaten

Zur Beschreibung der SISA-Ressourcen in GLASS wurde der in Unterabschnitt 5.2.1 auf Seite 83 vorgestellte *Dublin Core Metadata Element Set (DCMES)* verwendet (vgl. Tab. 5.7, Seite 116). DCMES

Zur Eingabe der insgesamt 15 Metadaten-Elemente des DCMES wurde die in Abb. 6.10 (Seite 181) dargestellte Web-Seite erstellt. Neben der Eingabe der Elemente des DCMES sind drei weitere Felder zu füllen: eins für den URN, eins für den Datei-Namen (falls es sich um eine Ressource handelt, deren Inhalt sich in einer Datei befindet) und eins für die Angabe einer Datei, in der die Metadaten gespeichert werden können. Die URN-Angabe und der evtl. eingetragene Dateiname werden durch die Web-Seite automatisch in das Element 'Identifier' übernommen. Eingabe

Die eingegebenen Metadaten können direkt in die Datenbank des Katalogmanagers übernommen werden. Die Verbindung der Web-Seite mit dem Eingesetzten DBMS (MySQL) geschieht, wie bereits in Unterabschnitt 6.3.1 ab Seite 172 beschrieben, über den Web-Server und PHP. Bei der Eingabe von Metadaten wird gleichzeitig auch die Ressourcen-Liste der Dokumentationskomponente aktualisiert. DBMS

Ist eine direkte Übertragung der Metadaten an den Katalogmanager nicht gewünscht oder nicht möglich (z. B. weil die Eingabe auf einem nicht vernetz- XML/ RDF

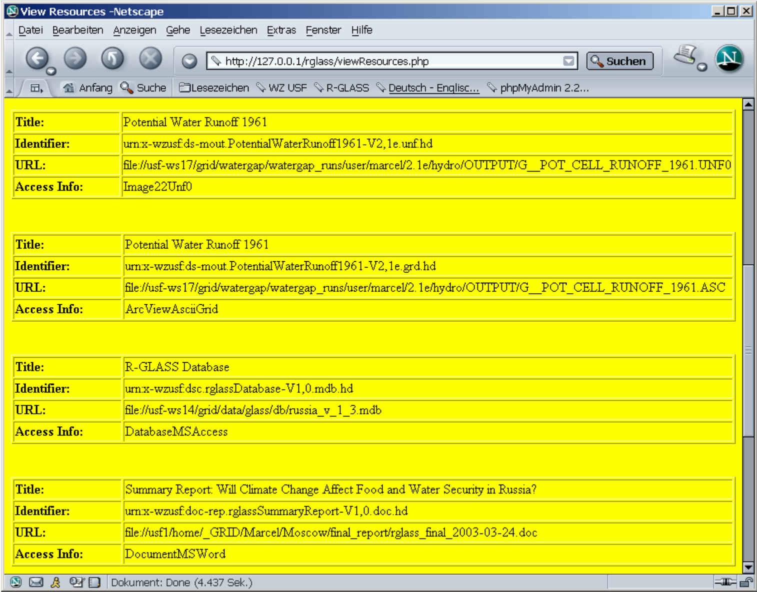


Abbildung 6.9: Anzeige der Ressourcen-Liste.

ten Rechner stattfindet), können die Metadaten auch in eine separate Datei geschrieben werden. Die Speicherung der Metadaten-Dateien erfolgt dann über die Kombination des *Resource Description Framework (RDF)* und der *Extensible Markup Language (XML)*[20], die an vielen Stellen für diese Zwecke gefordert wird (s. z. B. Anderson u. a., 2000). Der Inhalt einer solchen Metadaten-Datei ist in Anhang D.2 (Seite 247) kommentiert dargestellt.

XSL

Damit der Inhalt derartiger Metadaten-Dateien einfach und übersichtlich angezeigt werden kann, sind die Dateien mit einem so genannten 'XML-Stylesheet' (XSL) verknüpft. Ein solches Stylesheet enthält Anweisungen zur Transformation von XML in HTML; die Metadaten-Dateien können somit einfach in einem Web-Browser dargestellt werden.

Konvention

Um eine spätere Sammlung der Metadaten-Datei durch den im Rahmen des GLASS-Projektes entwickelten Metadaten-Sammler (s.u.) zu vereinfachen, enden die Namen der Dateien mit den Metadaten auf '.dc.xml'. Wird die Datei mit den Metadaten im gleichen Verzeichnis wie die Daten gespeichert, erhält

[20]XML ist eine Empfehlung des W3C. Startseite im Internet: `http://www.w3.org/XML`. Ausgiebige Erklärungen zu dieser Sprache und ihrer Anwendung finden sich z. B. bei Anderson u. a. (2000) u. Goldfarb und Prescod (2000).

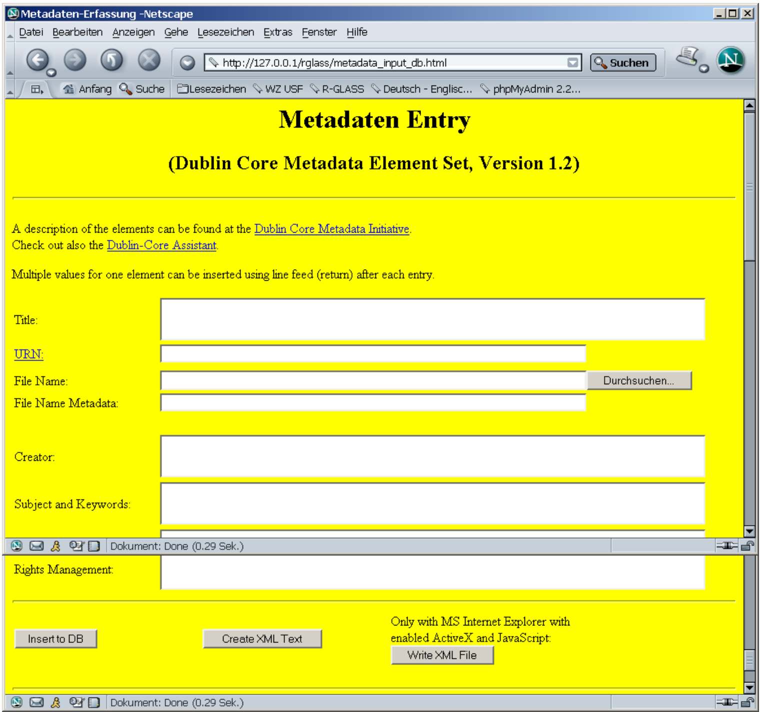


Abbildung 6.10: Web-Seite zur Erfassung der Metadaten nach dem *Dublin Core Metadata Element Set* (Ausschnitt des oberen und unteren Teils der Eingabemaske). Die Eingabe des Titels und einer URN ist obligatorisch für jede Ressource. Sofern es sich bei der beschriebenen Ressource nicht um eine 'abstrakte' Ressource handelt (s. Seite 74), muss zusätzlich der URL angegeben werden (z. B. file://meineRessource.txt). Die Seite bietet zwei grundsätzlich unterschiedliche Möglichkeiten der Metadaten-Speicherung: 1. die direkte Eintragung der Metadaten in die Datenbank der Katalogkomponente und 2. die Speicherung in einer XML/RDF-Datei.

der Metadaten-Sammler auch noch die Möglichkeit, den URL der Ressource (im 'file'-Schema, s. o.) automatisch den Metadaten hinzuzufügen.

6.3.2.4 Metadaten-Sammler

Die dezentrale Erfassung und Speicherung der Metadaten innerhalb der XML-Dateien hat den Vorteil, dass die Metadaten genau dort liegen, wo auch die Ressourcen selbst gespeichert sind. Damit alle Metadaten aber auch zentral Sammlung

eingesehen werden können, müssen die Einzelinformationen gesammelt und in einem zentralen Repository gespeichert werden. Für dieses ‘Einsammeln’ der Metadaten ist der *Metadaten-Sammler* (engl. metadata harvester) zuständig (vgl. Abb. 5.10, Seite 118).

Prinzip

Im Rahmen des GLASS-Projektes wurde ein einfacher Harvester implementiert und auf mehreren Rechnern installiert. Das Programm durchsucht zuvor eingestellte Verzeichnisse nach Metadaten (XML-Dateien mit der Endung ‘.dc.xml’), legt für die gefundenen Metadaten ein lokales Repository an und kopiert dieses – sofern gewünscht/möglich – anschließend an einen dafür vorgesehenen, zentralen Ort. Die auf diese Weise zusammengetragenen Informationen werden anschließend vom Katalogmanager in dessen eigene Datenbank integriert.

Umsetzung

Um die spätere Bearbeitung der Metadaten zu erleichtern, folgt die Speicherung des lokalen Repositorys ebenfalls dem XML/RDF-Format. Der Harvester wurde als Skript für die *Bourne-Again Shell (bash)* unter Unix (Linux) erstellt und benutzt die standardmäßig auf Unix-Maschinen verfügbaren Werkzeuge *awk* und *sed*.[21] Da die bash-Shell inklusive der Werkzeuge auch unter Microsoft-Windows verfügbar ist, kann der Harvester ohne Änderungen auch auf Rechnern mit diesem Betriebssystem installiert werden.[22]

Aktualisierung

Auf dem Projekt-Server von GLASS wurde der Harvester in die Liste der Programme aufgenommen, die regelmäßig vom System selbst aufgerufen werden.[23] Auf diese Weise ist gewährleistet, dass die Repositories täglich aktualisiert werden.

Integration

Die Integration der im XML-Format vorliegenden Repository-Daten in die Datenbank des Katalogmanagers wurde über ein PHP-Skript realisiert. Dieses Skript benutzt das *Document Object Model (DOM)* des W3C – eine plattform- und sprachneutrale Schnittstelle, um auf XML-Dokumente zuzugreifen und deren Inhalt zu manipulieren (s. z. B. Anderson u. a., 2000)[24] – und überträgt alle Metadaten des Repositorys über SQL in die Datenbank.

6.3.2.5 Metadaten-Anzeige

Stylesheet

Das über den Metadaten-Harvester erzeugte Repository besteht aus XML-Tags und kann daher auch direkt in einem Web-Browser angezeigt werden. Damit die Anzeige in einer ansprechenden Form erfolgt, wurde vom Harvester eine Stylesheet-Angabe in die XML-Datei geschrieben (s. Erklärungen zum Harvester im Anhang D.3, Seite 249). Durch das verwendete Stylesheet erfolgt die Anzeige im Web-Browser in Tabellenform. Abbildung 6.11 (Seite 184) zeigt die Anwendung des Stylesheets mit einem Eintrag aus dem automatisch erzeugten Metadaten-Repository für das R-GLASS-Projekt.

---

[21] *awk* ist ein Programm zur Bearbeitung von Textmustern, *sed* ein zeilenorientierter Texteditor. Nähere Informationen über diese Werkzeuge finden sich z. B. bei Herold (1999b). Eine

Die Metadaten, die sich direkt in der Datenbank des Katalogmanagers befinden, können über *phpMyAdmin* (s. o.) abgefragt, angezeigt und manipuliert werden. Datenbank

6.3.2.6 Automatische Metadaten-Generierung

Der verwendete Satz an Metadaten-Elementen (DCMES) beinhaltet ein Element namens ‘Source’. Der Wert dieses Elementes sollte eine Referenz auf eine Ressource beinhalten, die mit der beschriebenen Ressource in Beziehung steht (vgl. Tab. 5.7, Seite 116). Für Simulationsmodell-Ergebnisse enthält dieses Feld sinnvollerweise den Namen (URN) des verantwortlichen Simulationsmodells sowie den Namen (URN) des Simulationslaufes, in dessen Rahmen die Berechnung stattfand. Ein Blick in die Metadaten eines Simulationsmodell-Ergebnisses gibt dann direkt Aufschluss über dessen Herkunft (Simulationsmodell) sowie über die verwendeten Modelleinstellungen, da diese über den Simulationslaufnamen eindeutig bestimmt werden. Auf diese Weise kann die Berechnungsgrundlage für Ergebnisse transparenter und nachvollziehbarer gemacht und damit die Qualität eines Assessments erhöht werden. Beziehungen

Um die direkte Speicherung von Metadaten durch das Simulationssystem zu unterstützten, wird durch die Komponente der ‘allgemeinen Datenverarbeitung’ eine Funktion bereitgestellt, die Metadaten in der oben beschriebenen Form (DCMES und XML/RDF) als Datei speichern und Daten aus einer solchen Datei wieder lesen kann. Realisiert wurde diese Funktion durch eine in C++ implementierte Klasse, die von den neu erstellten Simulationsmodell-Teilen (vgl. Abb. 6.1, Seite 167) direkt verwendet wird. Eine Erweiterung dieser Klasse zu einem Dienst würde es ermöglichen, dass auch die nicht über C++ realisierten Teilmodelle die Funktionalität verwenden können. Klasse

6.3.3 Simulationssystem

6.3.3.1 Modellteile

Die Simulationssystem-Komponente ist verantwortlich für die Berechnung, Speicherung und Weitergabe von Simulationsergebnissen. Zur Erfüllung dieser Aufgaben wurden in Unterabschnitt 5.2.5 (Seite 131) zwei Schnittstellen

---

zusammenfassende Darstellung der Standard-Programme unter Unix findet sich bei Herold (1999a), ausführlichere Informationen hierzu geben Gulbins und Obermayr (1995).

[22]Getestet wurde der Harvester für MS-Windows-Rechner unter ‘Cygwin’ – einer Linux-ähnlichen Umgebung, die unter `http://www.cygwin.com/` kostenfrei zur Verfügung gestellt wird.

[23]Unter Unix können regelmäßig zu startende Programme in eine Tabelle (‘crontab’) aufgenommen werden. Für den Start dieser Programme sorgt der so genannte ‘cron’-Prozess des Systems (s. z. B. Welsh u. a., 2000).

[24]Die Spezifikation des DOM ist als ‘Recommendation’ auf den Internet-Seiten des W3C zu finden (`http://www.w3.org/DOM/DOMTR`).



title	**Will Climate Change Affect Food and Water Security in Russia?**
creator	Joseph Alcamo
creator	Genady Golubev
creator	Nikolai Dronin
creator	Andrei Kirilenko
creator	Marcel Endejan
subject	R-GLASS
subject	GLASS
subject	Russia
description	Will Climate Change Affect Food and Water Security in Russia?
description	Summary Report of the International Project on Global Environmental Change and its Thread to Food and Water Security in Russia
publisher	CESR
contributor	Guenther Fischer
date	2003-03-24
type	Document
type	Report
format	MS-Word
identifier	url:file://usf1/home/_GRID/Marcel/Moscow/final_report/rglass_final_2003-03-24.doc
identifier	urn:x-wzusf:doc-rep.rglassSummaryReport-V1,0.doc.hd
source	
language	en (also in Russian language available)
relation	
coverage	temporal: 1961-1990, 2020s, 2070s
coverage	spatial: 89 Russian Regions
rights	

title	**RGLASS Database**
creator	Marcel Endejan
subject	RGLASS

Abbildung 6.11: Web-Seite zur Anzeige von Metadaten.

definiert: eine zur Kontrolle von Simulationsläufen (ISimControl) und eine zur Abfrage von Ergebnissen (ISimResultAccess). Während die erste Schnittstelle zwei konkret zu realisierende Operationen enthält (init und run), besitzt die zweite Schnittstelle lediglich eine abstrakte Operation (getResult) – die genauen Operationen sind abhängig von konkreten Simulationsmodellen.

Basis-Klasse

Im Rahmen der GLASS-Entwicklung wurde eine C++-Klasse erstellt (SISA_ModelBase), die als Basisklasse für die neu erstellten Simulationsmodell-Teile dient und die einen Rahmen für die Kontrolle von Simulationsläufen bildet. Diese Basisklasse besitzt neben den Operationen *init()* und *run()* eine Ope-

ration, über die abgefragt werden kann, ob für eine bestimmte Simulationslauf-Spezifikation bereits Ergebnisse berechnet wurden: *resultsAvailable()*.

vier Dienste

Simulationsmodelle, die auf der Basis dieser Klasse realisiert sind, bieten vier generelle Dienste an: 1) die Abfrage, ob bestimmte Ergebnisse bereits berechnet wurden (AvailabilityRequest), 2) die Erzeugung von Ergebnis-Datensätzen für eine komplette Simulationszeit-Periode (DataSetGeneration), 3) die Abfrage bereits berechneter Ergebnisse (DataSetRequest) und 4) die Erzeugung neuer Ergebnisse zur Laufzeit (RunTimResult)[25].

Konkretisierung

Die Operationen zur Abfrage von Simulationsergebnissen sind abhängig von konkreten Simulationsmodellen. Für das Wasserstressmodell und das Nahrungsmittelstressmodell von GLASS wurde beispielsweise die Operation *getStress()* definiert, die für eine gegebene Region und einen gegebenen Zeitpunkt den Stresswert liefert. Abbildung 6.12 zeigt die Operationen der Basisklasse und des Nahrungsmittelstressmodels zur Verdeutlichung in Form eines UML-Klassendiagramms.

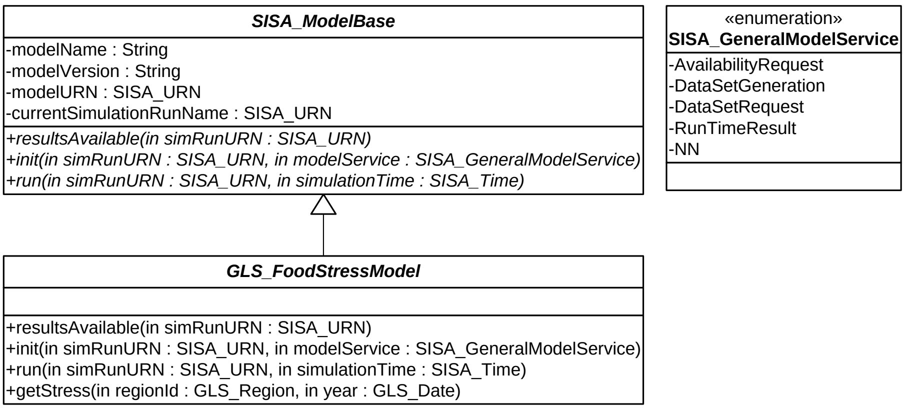


Abbildung 6.12: Basisklasse der Simulationsmodelle. Die Basisklasse der Simulationsmodelle (*SISA_ModelBase*) enthält die virtuellen Operationen *resultsAvailable()* (zur Abfrage, ob ein Simulationslauf bereits durchgeführt wurde), *init()* (zur Initialisierung eines Dienstes) und *run()* (zum Starten der Berechnung neuer Ergebnisse). Das von der Basisklasse abgeleitete Simulationsmodell zur Berechnung des Nahrungsmittelstresses (*GLS_FoodStressModel*) realisiert die rein virtuellen Funktionen der Basisklasse und fügt eine weitere zur Abfrage von Ergebnissen hinzu: *getStress()*. Die Umsetzung des Wasserstressmodells erfolgt analog.

[25] Da oft aufwendige Initialisierungen zur Berechnung eines Wertes notwendig sind, ist die direkte Berechnung eines einzelnen, angefragten Wertes nicht immer sinnvoll bzw. möglich.

Adapter

Simulationsmodelle, die aus anderen Projekten übernommen und nicht neu erstellt werden, bieten die geforderten Schnittstellen und Operationen i. d. R. nicht direkt an – das WaterGAP-Modell hat beispielsweise keine Operation namens 'init()'. Sofern diese Modelle die geforderten Funktionen über andere Schnittstellen bzw. Operationen bereitstellen, müssen die Schnittstellen entsprechend angepasst werden. Diese Anpassung kann mit Hilfe so genannter *Adapter (Wrapper)* erfolgen. Adapter nutzen die verfügbaren Operationen des anzupassenden Objektes und stellen mit deren Hilfe die geforderten Operationen zur Verfügung. Abbildung 6.13 verdeutlicht das Prinzip in Form eines UML-Sequenzdiagramms.

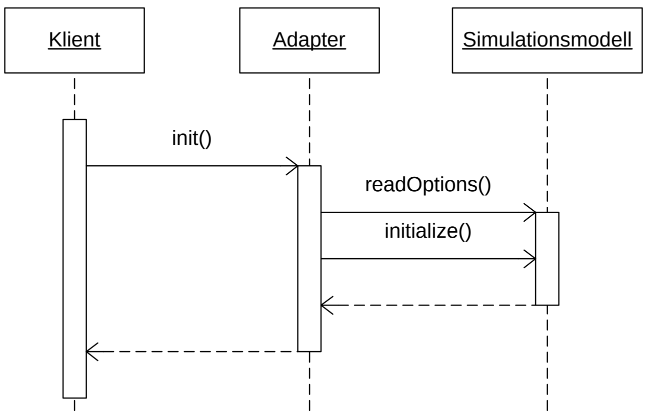


Abbildung 6.13: Prinzip eines Adapters. Laut *SISA-Schnittstellendefinition* besitzt ein Simulationsmodell die Operation *init()*. Das beispielhafte *Simulationsmodell* besitzt diese Operation aber nicht und benötigt zur Initialisierung den Aufruf von zwei anderen Operationen: *readOptions()* und *initialize()*. Damit ein Client dennoch die im SISA definierte Schnittstelle mit der Operation *init()* für dieses Modell benutzen kann, sorgt der *Adapter* für eine transparente Umsetzung auf die entsprechenden Operationen des Modells – der Adapter passt die Schnittstelle des Simulationsmodells damit an die geforderte SISA-Schnittstelle an. Weitergehende Informationen über das Adapter-Muster finden sich z. B. bei Gamma u. a. (1996).

Erweiterung

Bietet ein Simulationsmodell keine Operationen, die zur Anpassung an die geforderten Schnittstellen verwendet werden können, so ist das Modell entsprechend zu modifizieren oder zu erweitern. Eine Implementierung der Operation *run()* kann dabei als Minimalanforderung zur 'Integration' eines Modells in ein SISA betrachtet werden.

Das Wasserstressmodell und das Nahrungsmittelstressmodell wurden direkt in C++ geschrieben und sind Ableitungen der abstrakten Basisklasse *SISA_ModelBase*. Für das WaterGAP-Modell wurde ein einfacher – ebenfalls auf der SISA_ModelBase basierender – Adapter geschrieben, der nach Aufruf der Operation *run()* einen Modelllauf über die gesamte Simulationszeit startet.

6.3.3.2 Modellkopplung

In der derzeitigen Version des GLASS-Modells sind die Teilmodelle einfach miteinander verkettet. Die Ausgabedaten eines Modells werden als Eingabedaten für ein nachgeordnetes Modell verwendet, Rückkopplungen zwischen den Modellen gibt es nicht (vgl. Abb. 6.1, Seite 167). Die Teilmodelle schreiben ihre Ergebnisse in Dateien, die im Anschluss von den nachgeordneten Modellen wieder gelesen werden (bzw. über die Schnittstelle *ISimResultAccess* direkt vom Modell bezogen werden). Verkettung

Die neu erstellten (und nicht verteilten) Modellteile von GLASS (u. a. das Wasserstress-, das Nahrungsmittelstress- und das Sicherheitsmodell) können über einen einfachen *Simulationsmodell-Manager* integriert werden, der eine Liste aller beteiligten Teilmodelle enthält und auf Anfrage Referenzen auf diese Modelle zurückgibt. Der Eintrag in die Liste wird von den Teilmodellen selbst (durch eine Anmeldung) veranlasst. Eine Koordination der Zeitschritte oder die Verwaltung gemeinsam genutzter Variablen ist aufgrund der Kaskadierung der Modelle nicht notwendig. Modell-Manager

6.3.4 Simulationslaufmanager

Der Simulationslaufmanager ist verantwortlich für die Verwaltung der simulationslaufspezifischen Einstellungen und die Bereitstellung dieser Informationen für die Simulationssystem-Komponente (s. Unterkapitel 5.2.4, Seite 128).

Die Simulationsmodelle innerhalb der Simulationssystem-Komponente können verteilt über mehrere Hosts realisiert sein. Im Falle des GLASS-Modells laufen die Teilmodelle auf unterschiedlichen Betriebssystemen und das Gesamtmodell ist alleine aus diesem Grund bereits ein verteiltes System. Um die Verwaltung der simulationslaufspezifischen Modelleinstellungen zu vereinfachen und um Ergebnisse zu einem späteren Zeitpunkt besser nachvollziehen zu können, bietet sich für den Simulationslaufmanager allerdings eine nicht-verteilte Realisierung an. Die Konzentration *aller* simulationslaufspezifischen Einstellung in einem zentralen Simulationslaufmanager erhöht darüber hinaus die Konsistenz von Simulationsläufen – eine nicht passende Kombination der Einstellungen unterschiedlicher Modellteile fällt bei einer zentralen Verwaltung eher auf und lässt sich besser (auch formal) überprüfen. Darüber hinaus ermöglicht die zentrale Verwaltung der Simulationslaufspezifikationen diejenigen Einstellungen nur einmal zu speichern und zu spezifizieren, die für mehrere verteilt vs. zentral

Teilmodelle gültig sind (z. B. den Start- und Endzeitpunkt des Simulationszeitraums).

Prinzip

Im Rahmen des GLASS-Projektes wurde der Simulationslaufmanager aus den genannten Gründen als Server realisiert. Dieser Server verwaltet alle simulationslaufspezifischen Einstellungen und bietet die Einstellungen auf Anfrage den Simulationsmodellen (Clients) an.

Umsetzung

Sockets

Zur Kommunikation zwischen Client und Server, die auch auf *einem* Host laufen können, wurden die so genannten 'Sockets' in Verbindung mit dem *Transmission Control Protocol/Internet Protocol (TCP/IP)* benutzt. Abbildung 6.14 gibt einen Überblick über das Prinzip dieser Kommunikationsart. Die Kommunikation über Sockets und TCP/IP wurde gewählt, da sie relativ einfach realisierbar ist und da TCP/IP auf vielen Plattformen bereitsteht – TCP/IP ist das Protokoll, das am häufigsten zur Kommunikation in lokalen und weltweiten Netzen (z. B. dem Internet) eingesetzt wird (Herold, 1999a).

Adressierung

Die Adressierung zwischen Client und Server geschieht über die *Internet-Adresse (IP-Adresse)* der Hosts und über die Nummer des verwendeten *Kommunikationskanals (Port)* auf dem Host. Programme, die mit dem Simulati-

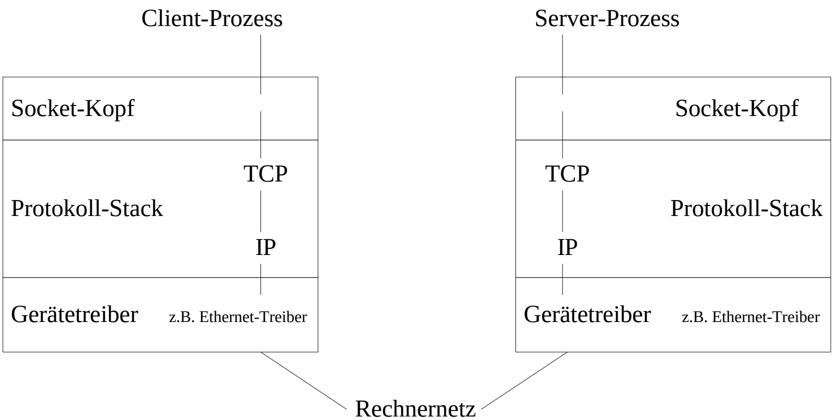


Abbildung 6.14: Kommunikation über Sockets. Die Abbildung zeigt das Socket-Modell am Beispiel von TCP/IP. Die Sockets sind die 'Datenendpunkte' zur Kommunikation zwischen Prozessen. Sie bauen auf dem *Transmission Control Protocol (TCP)* auf, das wiederum auf dem *Internet Protocol (IP)* fußt. Die 'unterste' Schicht, der Gerätetreiber, sorgt schließlich für die Kommunikation über das Rechnernetz. Client und Server nutzen (sehen) lediglich die Funktionen des Socket-Kopfes, die anderen Schichten sind transparent. Quelle der Abbildung: Gulbins und Obermayr (1995).

onslaufmanager kommunizieren wollen, müssen lediglich die IP-Adresse seines Hosts und den Port des Managers kennen, um mit ihm in Verbindung zu treten.

Um die Kommunikation der in GLASS eingesetzten C++-Programme mit dem Server zu vereinfachen, wurde eine Klasse (SISA_SimulationRunManagerClient) erstellt, die die Kommunikation über die Sockets transparent macht. In C++ geschriebene Modellteile rufen also lediglich eine Methode dieser Klasse auf und brauchen sich nicht um die Kommunikation mit dem Server zu kümmern.[26] Der SISA_SimulationRunManagerClient kontaktiert bei einer Anfrage automatisch den Server, erfragt den Wert der gewünschten Einstellung und gibt ihn anschließend an das Simulationsmodell weiter. Client

Der Server, der ebenfalls über eine C++-Klasse (SISA_SimulationRunManagerServer) implementiert ist, bezieht alle simulationslaufspezifischen Informationen aus einer XML-Datei. Da Simulationsläufe als abstrakte Ressourcen betrachtet werden, können die zugehörigen Einstellungen innerhalb des SISA eindeutig identifiziert werden. Innerhalb des Servers werden die Einstellungen über das *Document Object Model (DOM)* abgebildet. Der Server stellt den Clients auf Anfrage einzelne Einstellungen in Form von Zeichenketten zur Verfügung. Abbildung 6.15 (Seite 190) verdeutlicht das Zusammenspiel von Client und Server noch einmal. Server

Die Verwendung des XML-Formats bietet gegenüber anderen Formaten den Vorteil, dass sowohl die Strukturen als auch die gültigen Werte von Simulationsmodell-Spezifikationen auf eine einfache und vor allem standardisierte Weise definiert werden können. Diese Definitionen können entweder über die *Document Type Definition (DTD)* oder über die *XML Schema Description Language (XSD)* erfolgen. Der Simulationslaufmanager ist auf diese Weise in der Lage, die Einstellungen beim Laden der Datei auf ihre Gültigkeit hin zu überprüfen. Die Überprüfung selbst ist Sache eines validierenden XML-Parsers,[27] dem neben der XML-Datei die entsprechende DTD- bzw. XSD-Datei zur Verfügung gestellt wird. Wird dem Server eine nicht gültige Simulationsmodell-Spezifikation übergeben (z. B. mit einer fehlenden Variablen-Belegung), so erkennt er beim Einlesen der Spezifikation den Fehler und kann direkt eine entsprechende Meldung ausgeben. Über XML-Schemata ist es darüber hinaus möglich den Wertbereich für Variablen festzulegen. Ein Simulationslauf mit Modelleinstellungen, für die Simulationsmodelle nicht ausdrücklich vorgesehen oder getestet wurden (z. B. die Unterschreitung des minimalen Zeitschrittes), kann auf diese Weise unterbunden werden. Validierung

[26] Entsprechende Funktionen sollten für alle in der Simulationssystem-Komponente verwendeten Programmiersprachen bereitgestellt werden.

[27] Ein Parser ist ein Programm, das einen Quelltext in seine einzelnen Bestandteile zerlegt und das überprüfen kann, ob der Quelltext syntaktisch korrekt ist (s. Engesser, 1993). Validierende XML-Parser prüfen sowohl Form als auch Inhalt von XML-Quelltexten (s. Anderson u. a., 2000).

Beispiel

Ein kommentiertes Beispiel für den Einsatz des Simulationslaufmanagers ist, zusammen mit Erklärungen zur Realisierung der nachfolgend beschriebenen Datenzugriffskomponente, im Anhang D.4 (Seite 250) zu finden.

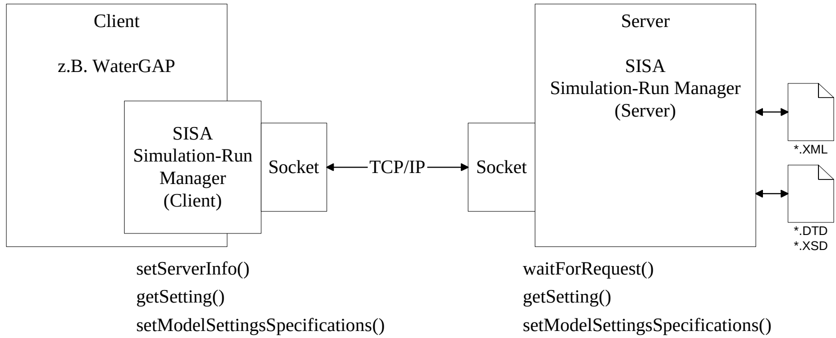


Abbildung 6.15: Realisierung des Simulationslaufmanagers. Der Simulationslaufmanager ist als Server realisiert. Die Simulationslauf-Spezifikationen sind in Form von XML-Dateien (in Verbindung mit einer DTD/XSD-Datei) gespeichert. Die Auswahl einer solcher Datei erfolgt über die Methode *setModelSettingsSpecification()*. Beim Laden der XML-Datei können die in ihr enthaltenen Einstellungen unter Zuhilfenahme einer DTD/Schema-Datei validiert werden. Nach dem Aufruf der Methode *waitForRequest()* wartet der Server auf Anfragen von Clients. Die Kommunikation geschieht über den TCP/IP-Protokoll-Stack (Client und Server kommunizieren über die *Sockets* der Hosts, auf denen sie laufen). Der Client kann entweder direkt mit dem Server kommunizieren oder – wie dargestellt – die hierfür geschriebene Klasse *SISA_SimulationRunManagerClient* benutzen. Nachdem der Client über *setServerInfo()* einen Server ausgewählt hat (IP-Adresse und Port), kann er – unter Angabe des Simulationslaufnamens – über *getSetting()* einzelne Modelleinstellungen abfragen. Sofern notwendig kann der Client zunächst die zu verwendende XML-Datei mit den Einstellungen festlegen (über *setModelSettingsSpecification*).

6.3.5 Datenzugriff und Datenbasis

6.3.5.1 Datenzugriff

Die Datenzugriffskomponente ist verantwortlich für den transparenten lesenden und schreibenden Zugriff auf Daten und die Transformation zwischen Datenformaten (s. Unterabschnitt 5.2.6, Seite 134).

Die Datenzugriffskomponente kann als Vermittler von Datensätzen angesehen werden: Clients (z. B. Simulationsmodelle) wenden sich unter Angabe der Datensatzbezeichnung an den Vermittler und erhalten von ihm den gewünschten Datensatz, der sich an einem beliebigen, durch den Vermittler zugreifbaren Ort befindet.

Bei dieser Art des Datenzugriffs handelt es sich um eine Schichten-Architektur: der Client greift auf die Datenzugriffskomponente zu und diese wendet sich an die Datenhaltungsschicht bzw. an die Datenbasis (vgl. Abb. 5.25, Seite 136). Schichten

Mediatoren

Der Zugriff auf Datensätze ist für den Nutzer transparent: Einzelheiten zu Ort und Format der *Speicherung* des Datensatzes sollten für den Nutzer irrelevant sein. Die Datenzugriffskomponente verbirgt diese Einzelheiten und transformiert das Format eines gespeicherten Datensatzes gegebenenfalls in das gewünschte Format. Innerhalb eines SISA sind i. d. R. viele Datenformate zu berücksichtigen. Die Vermittlung (Mediation) zwischen den einzelnen Formaten innerhalb der Datenzugriffskomponente sollten aus Gründen der Erweiterbarkeit und Flexibilität modular aufgebaut sein. Shaw und Garlan (1996) schlagen hierzu das Prinzip der 'Mediatoren' vor.[28] Die Schicht zwischen Nutzer-Funktionen und Datenbasis besteht hiernach aus einzelnen Mediatoren, die für den Zugriff auf verschiedene Datenquellen zuständig sind. Abbildung 6.16 (Seite 192) verdeutlicht dieses Prinzip in graphischer Form. Mediator

Im Rahmen der GLASS-Entwicklung wurden Mediatoren für die wichtigsten Datenformate realisiert. Ein besonderer Schwerpunkt lag dabei auf der Integration des proprietären Datenformats des Modells IMAGE2.2, das auch im WaterGAP-Modell verwendet wird. Bei diesem Format handelt es sich um ein unformatiertes Binärformat (Abkürzung 'UNF') für vier verschiedene Datentypen: Ganzzahlen in 8 Bit-, 16 Bit- und 32 Bit-Darstellung (abgekürzt UNF1, UNF2, UNF4) sowie Fließkommazahlen in 32 Bit-Darstellung (UNF0). Ein Datensatz im UNF-Format von IMAGE2.2 besteht aus insgesamt 66896 Werten. Jeder Wert repräsentiert dabei einen Teil der gesamten Landoberfläche der Erde, genauer: eine 0.5° geographischer Länge * 0.5° geographischer Breite große Rasterzelle. Die Georeferenzierung der Zellen, also die Zuordnung einer eindeutigen Position auf der Erdoberfläche für jede Zelle, geschieht über die Angabe von Zeilen (1-720) und Spalten (1-360), die in zwei weiteren Datensätzen gespeichert sind. UNF-Format

Mit der Verwendung des UNF-Formats gehen einige Probleme einher: so können beispielsweise nur Werte für Zellen gespeichert werden, die auch in der Landmaske von IMAGE2.2 auftreten; Änderungen in der Landmaske ziehen UNF-Probleme

[28] Nicht zu verwechseln mit dem Mediator-Muster von Gamma u. a. (1996).

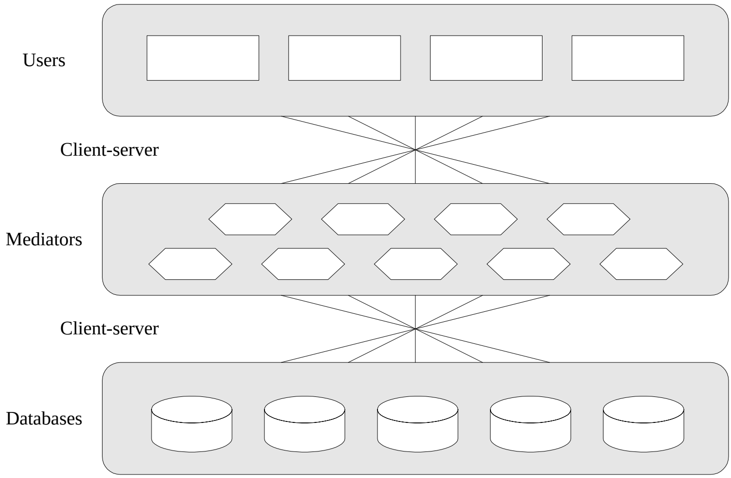


Abbildung 6.16: Prinzip der Mediatoren. Mediatoren sind Teile einer hierarchischen Schichtenarchitektur, in der die Funktionen der Nutzer (Users) getrennt sind von den Datenbanken (Databases). Die Mediatoren vermitteln dabei zwischen den Anfragen der Nutzer und den (heterogenen) Datenbanken. In der SISA-Architektur entspricht die Datenzugriffskomponente einer solchen Mediator-Schicht. Quelle der Abbildung: Shaw und Garlan (1996).

u. U. Änderungen der Programme nach sich[29]; aus dem Datenformat ist nicht ersichtlich, ob es sich um vorzeichenbehaftete oder vorzeichenlose Daten handelt[30]; und durch die binäre Speicherung und der damit verbundenen Frage der Binärkodierung der Daten (little endian oder big endian) sind Datensätze schlecht portierbar[31].

Für das UNF-Format wurde aus diesen Gründen eine Reihe von Mediatoren implementiert. Zu den wichtigsten Mediatoren gehören jene zur Vermittlung zwischen den Formaten UNF0/1/2/4 und

[29] IMAGE2.1 enthielt beispielsweise nur 59831 Zellen. Programm-Schleifen über alle Zellen konnten daher über 16-Bit-Zahlen realisiert werden – für IMAGE2.2 ist das nicht mehr möglich (der maximal darstellbare Wert für eine ganzzahlige 16-Bit-Variable ist 65535).

[30] Die Formatangabe 'UNF2', die auch als Dateiendung verwendet wird, bezieht sich auf den Wertebereich von 0-65535 *oder* auf den von -32768 bis 32767.

[31] Die Wertigkeit der Bytes innerhalb eines Datenwortes unterscheidet sich zwischen Plattformen: während Intel-basierte Systeme das niederwertigste Byte einer binären Zahlenrepräsentation zuerst schreiben (little endian), verfolgen beispielsweise die Systeme von SUN-Microsystems unter SPARC sowie Hewlett-Packard-Maschinen unter HP-UX den umgekehrten Weg und gehen davon aus, das höchstwertige Byte zuerst (an der niedrigsten Adresse) im Speicher vorzufinden (big endian).

- dem ASCII-Format des GIS 'ArcView'
- Tabellen in relationalen Datenbanken

Das ArcView-ASCII-Format beschreibt einen rasterbasierten Geodatensatz über ein regelmäßiges Raster und je einem Wert pro Rasterzelle. Die Daten sind in Text-Form im ASCII-Format[32] gespeichert und somit auch zwischen unterschiedlichen Plattformen gut austauschbar und mit einfachen Mitteln (z. B. einem Texteditor) darstellbar. Zusätzlich zu den Daten enthält jede Datei dieses Formats in den ersten Zeilen eine kurze Beschreibung der Charakteristiken des Datensatzes (z. B. die geographische Ausdehnung einer Zelle). Das Format des Datensatzes wird im Anhang D.6 (Seite 257) kurz erklärt. ASCII-Format

Die Vermittlung zwischen dem UNF-Format und den Tabellen relationaler Datenbank-Managementsysteme (RDBMS) wurde über die *Open Database Connectivity (ODBC)* realisiert. ODBC ist eine auf Treibern basierende Schnittstelle zum herstellerunabhängigen Zugriff auf Datenquellen und wird von allen namhaften Datenbank-Herstellern unterstützt.[33] Die ODBC-Datenbank-Mediatoren wurden erfolgreich getestet für das RDBMS *MySQL* sowie für das System *Microsoft Access 2000*. ODBC

Aufgrund der relativ schlechten Performance beim Zugriff auf Datensätze über ODBC wurden ebenfalls Mediatoren implementiert, die auf den *Active Data Objects (ADO)* von Microsoft basieren und die einen schnelleren Zugriff auf die Daten erlauben.[34] ADO

Zum Export von UNF-Datensätzen wurden weitere Mediatoren realisiert. Diese erlauben die Speicherung von Datensätzen in den Formaten weitere Formate

- ArcView Grid
- Geography Markup Language (GML)
- GeoTiff
- ASCII-Tabelle

Das Format *ArcView Grid* ist ein proprietäres Rasterformat des GIS *ArcView* des Herstellers ESRI. Die *Geography Markup Language (GML)* ist ein Standard des OGC (Cox u. a., 2003) – der nicht nur zur Kodierung von Rasterdaten eingesetzt werden kann – und GeoTiff ein Rasterformat, das das für Bilddateien benutzte Tiff-Format um geographische Informationen erweitert. Das ArcView-Format wurde gewählt, da dieses GIS im Rahmen des GLASS-Projektes eingesetzt wird. Die Möglichkeit des Exports von Daten in die Formate GML und GeoTiff wurde implementiert, da diese Standards eine breite Anwendung versprechen (s. z. B. Kuhn u. a., 2001). Neben diesen Formaten

[32] *American Standard Code for Information Interchange (ASCII)* – ein auf vielen Rechnern verwendeter 7-Bit-Code zur Darstellung von Ziffern, Buchstaben und Sonderzeichen.

[33] Zur Verwendung von ODBC unter Oracle s. z. B. Herrmann u. a. (1998).

[34] Nähere Informationen zu ADO finden sich z. B. bei Gordon (2000).

kann ein UNF-Datensatz auch als Tabelle im ASCII-Format abgespeichert werden, die anschließend einfach in andere Programme (z. B. in DBMS) importiert werden kann.

GIS-Kopplung

Der Mediator zum Export in das ArcView-Grid-Format nutzt die Programmierschnittstelle von ArcView und stellt somit ein einfaches Beispiel für die Kopplung der Komponente mit einem externen GIS dar.

Binärkodierung

Neben den angeführten Mediatoren gibt es solche, die einen transparenten Zugriff auf UNF-Dateien erlauben, indem sie eine automatische Konvertierung der beiden möglichen Binärkodierungen vornehmen.

Realisierung

Realisierung

Die Komponente für den Datenzugriff wurde als C++-Klasse implementiert und läuft derzeit als Server auf dem Betriebssystem *Microsoft Windows 2000*. Die Kommunikation zwischen Client und Server geschieht über den bereits in Unterabschnitt 6.3.4 (Seite 187) erklärten Mechanismus der *Sockets*. Für die Seite des Client wurde – wie beim Simulationslaufmanager (vgl. 6.3.4, Seite 187) – eine C++-Klasse zur Kapselung der Datenübertragung implementiert. Ein

Beispiel

Beispiel zur Verwendung der Datenzugriffskomponente findet sich im Anhang D.4 (Seite 250).

6.3.5.2 Datenbasis

Basis

Die im Rahmen des GLASS-SISA realisierten Mediatoren beschränken sich derzeit auf Geodaten im IMAGE2.2-Format. Diese Daten können über die Mediatoren sowohl in Form von Dateien in unterschiedlichen Formaten als auch innerhalb eines RDBMS, d. h. des Datenbanksystems, gespeichert werden. Aufgrund der eingeschränkten Verfügbarkeit von Mediatoren müssen die Simulationsmodelle aber auch direkt – also an der Datenzugriffskomponente vorbei – auf Dateien zugreifen.[35]

Erweiterung

Um die Integration der Datensätze in das SISA zu verbessern und das Speichern *eines* Datensatzes in unterschiedlichen Formaten zu vermeiden, sind die Mediatoren entsprechend zu erweitern. Die Erweiterung der Mediatoren sollte neben dem Geodaten-Format von IMAGE2.2 auch andersartige Geodaten sowie Daten in Form von Listen und Tabellen behandeln können.

Formate

Bei der Realisierung und Erweiterung der Mediatoren stellt sich die Frage der Speicherungsart (Datei vs. Datenbanksystem) sowie den zu verwendenden Datenformaten für die Speicherung. In dieser Frage konnte sich (noch) kein Standard durchsetzen: Daten, die zur Initialisierung von Modellen benutzt werden, sind heterogen und „Wissenschaftler sind verwirrt“ über die verschiedenen

[35] Eine ‘Integration’ der Daten in das System erfolgt dennoch: durch die eindeutige Bezeichnung der Datensätze (Ressourcen) und die Bereitstellung der Zugriffsinformationen durch den Katalogmanager.

Datentypen, Formate und Systeme im Umkreis wissenschaftlicher Daten im Rahmen der Erdsystemforschung (Ramachandran u. a., 2003).[36]

Nicht nur im Rahmen der Speicherung und Übertragung von Geodaten kristallisiert sich die Verwendung des XML-Formats als möglicher Standard heraus (s. GML, Cox u. a., 2003). Dieses Format hat weitere Vorteile, die Anderson u. a. (2000) wie folgt zusammenfassen: XML

- gute Archivierungsmöglichkeit der Daten
- leichter Austausch von Daten
- die Daten sind über das Document Object Model (DOM) auch von einfachen Clients, z. B. einem Web-Browser, zu bearbeiten
- flexible Darstellungsmöglichkeit der Daten, z. B. über Stylesheets
- möglicher Import und Export von Daten in bzw. aus Datenbanken[37]

Auch für die Archivierung von Daten bietet XML nach Anderson u. a. (2000) gegenüber anderen Datenformaten, wie einfachen Dateien oder Datenbank-Dumps, Vorteile:

- XML-Dateien sind plattformunabhängig – einfache Bearbeitung durch DOM-Parser möglich
- XML-Dateien sind selbstbeschreibend – Strukturinformationen sind inhärent; nur wenig Zusatzinformation zum Verständnis eines Dokuments notwendig
- XML-Dateien beschreiben hierarchische Informationen – einfaches Verständnis der Daten durch Baumstruktur

Der oft angeführte Nachteil der umfangreichen Größe von XML-Dateien wird ebenfalls durch Anderson u. a. (2000) entkräftet: da es sich um einfache Text-Dateien handelt, können diese i. d. R. durch Komprimierung auf ein Zehntel oder Zwanzigstel der ursprünglichen Größe reduziert werden. Als Nachteil dieses Formats bleibt allerdings eine im Vergleich zu anderen Arten der Datenspeicherung schlechtere Auswertungsgeschwindigkeit – XML-Daten müssen vor der Verwendung innerhalb eines Programms zunächst durch einen Parser in das Document Object Model überführt werden.

Die derzeitigen Aktivitäten im Bereich der GML und die beginnende Unterstützung dieses Formats durch GIS-Hersteller sprechen ebenfalls für die Verwendung von XML. Darüber hinaus erlaubt die Speicherung von Datensätzen im XML-Format eine *direkte Integration* von Metadaten in den Datensatz.

[36] Einer der Gründe, warum eine automatisierte Transformation von Datentypen notwendig ist.

[37] Siehe hierzu z. B. die Informationen zur XML-Datenbank 'Xindice' unter `http://xml.apache.org/xindice`

6.3.5.3 Datenbanksystem

Als Datenbanksysteme wurden, wie bereits erwähnt, *MySQL* und *Microsoft Access* eingesetzt. Der Einsatz der Systeme beschränkt sich bisweilen auf den Zugriff durch die angesprochenen Mediatoren.

6.3.5.4 Datengrundlage

Neben der Verwendung eines 'allgemein anerkannten' *Formats* von Datensätzen, ist es im Rahmen der Forschung zum System Erde wichtig, auf anerkannte *Inhalte* zurückzugreifen. Diese Notwendigkeit begründet sich u. a. darauf, dass zur Validierung von Modellen deren Ergebnisse mit den Ergebnissen anderer Modelle verglichen werden sollen (Toth, 1995).

anerkannte Quellen

Die im Rahmen der Systemdefinition aufgelisteten grundlegenden Assessment-Daten (/D60/-/D120/) sollten daher aus einschlägig bekannten und anerkannten Quellen (UN-Organisationen, IPCC, Weltbank etc.) kommen.

Standardisierungsbemühungen

In diesem Zusammenhang sind auch die Bestrebungen des *International Steering Committee for Global Mapping (ISCGM)*[38] von Bedeutung, das die Erstellung von Geodaten mit einer Auflösung von 1km * 1km und globaler Abdeckung koordiniert. Diese so genannte *Global Map* besteht aus acht Schichten (Themen): Grenzen, Entwässerung, Transport, Siedlungen, geographische Höhe, Landbedeckung, Landnutzung, Vegetation.[39] Die Verwendung derartiger Basisdaten würde die Vergleichbarkeit von Analysen und – aufgrund der dann einheitlichen Landmaske – auch die Interoperabilität von Modellen erheblich verbessern.[40]

6.3.5.5 Kodierungsstandards

Der Austausch von Ressourcen macht nicht bei den Modellen oder Modellteilen bzw. Diensten Halt. Auch Daten, deren Erzeugung oder Beschaffung sehr arbeitsaufwendig ist, stellen eine wichtige Arbeitsgrundlage beim integrierten Assessment dar, deren Wiederverwendung anzustreben ist. Der Austausch von Daten funktioniert aber nur dann reibungslos, wenn sie nicht nur die gleiche Form im Sinne der verwendeten Datenelemente (Syntax) und die gleiche Inhaltsbedeutung (Semantik) besitzen, sondern darüber hinaus auf die gleiche Art und Weise kodiert wurden: Entfernungsangaben in Meilen mit einem Punkt als Dezimaltrennzeichen lassen sich nicht ohne Umwandlungsaufwand in Systemen verwenden, die die Angaben in Metern und einem Komma als Dezimaltrennzeichen benötigen. Im Sinne der Interoperabilität ist daher die Verwendung

[38] Startseite im Internet: `http://www.iscgm.org/html4/index.html`

[39] Die Spezifikation der *Global Map* ist in der Version 1.1 erhältlich unter `http://www.iscgm.org/html4/pdf/gmspec-1.1.pdf`

[40] Siehe zu diesem Thema auch das *Spatial Data Infrastructure Cookbook* von Nebert (2001).

einheitlicher Metriken und Kodierungen anzustreben. Ausgangspunkte hierfür sind erneut Standards.

SI-Einheiten

Als Grundlage sollten die allgemein bekannten SI-Einheiten dienen. Die Verwendung dieser Einheiten mag in einigen Fällen (und Ländern) zunächst umständlich und wenig eingängig erscheinen: man denke an einen Datensatz, der eine mittlere Monatstemperatur von 280 anzeigt – Kelvin, wohlgemerkt. Temperaturangaben in Kelvin haben allerdings den Vorteil, dass sie stets als positive Zahlen dargestellt werden (einfachere Datenhaltung und -validierung) und dass diese Metrik in vielen wissenschaftlichen Gleichungen verwendet wird.

Gerade beim internationalen Datenaustausch stellen Standards und andere Vereinbarungen nicht nur eine große Arbeitserleichterung dar, sondern auch einen Gewinn bezüglich der Richtigkeit bei der Verwendung von Daten. Die Angabe '320 centner/ha', die aus einer russischen Statistik über Ernte-Erträge entnommen wurde, ist beispielsweise nicht direkt vergleichbar mit derselben Zahl aus einer deutschen Statistik – in Russland bedarf es 100 kg für einen Zentner, in Deutschland sind es 50 kg (100 Pfund). Selbst bei einem Datensatz mit dokumentierter Einheit würde ein solches Problem nicht direkt (wenn überhaupt) auffallen.

ISO-Normen

Datum

Neben 'einheitlichen' SI-Einheiten gibt es Standards (insbesondere der ISO-Serie) zur Implementierung und Formatierung von Daten. Ein Beispiel für deren Sinnhaftigkeit ist die Angabe eines Datums: '08/05/03' – in den USA würde diese Angabe dem 5. August 2003 entsprechen, in Deutschland dem 8. Mai 2003. Eine Mehrdeutigkeit trotz eines wohldefinierten Formats, denn '2003-05-08'[41] ist die standardkonforme Art für die numerische Notation des 8. Mai 2003 nach ISO 8601. Diese Norm regelt über die Datums-Angaben hinaus auch die Angabe von Wochennummern und Zeiten: 11:55:00Z entspricht fünf vor zwölf gemessen am Null-Meridian (Greenwich).

Länder

Ein im Rahmen der globalen Modellierung immer wieder aufkommendes Problem ist die uneinheitliche Kodierung von Ländern. Bei der Realisierung des GLASS-Modells mussten für eine Analyse beispielsweise drei Datensätze unterschiedlicher Datenlieferanten miteinander verknüpft werden. Die Bezeichnungen für die USA lauteten in diesen Datensätzen wie folgt: 'United States', 'U.S.', 'America, US' – an eine automatische Zusammenführung der Datensätze war (nicht nur wegen der Länderkennung der USA) nicht zu denken. Für GLASS wurden allen länderbezogenen Datensätzen eindeutige Länderkennungen zugewiesen. Hierzu wurde die aus drei Ziffern bestehende Kennung nach ISO 3166 verwendet. Diese Norm erlaubt neben der Kodierung existierender Länder auch die Kodierung nicht mehr existierender Staaten (wie der DDR) und administrativer Einheiten, die für manche Assessments relevant sind.

[41] Formal: JJJJ-MM-TT. Die Bindestriche können auch entfallen. Ist die Angabe des Tages oder Tages und Monats nicht gewünscht, wird '2003-05' bzw. '2003' geschrieben.

Die angesprochenen Kodierungen sind nur Beispiele. Tabelle 6.1 listet weitere relevante Standards auf.

Standard	Gegenstand	Kurzbeschreibung
ISO 2955	Das metrische System	Methode zur Beschreibung von SI-Einheiten und anderen Maßen in Computersystemen.
ISO 3166	Ländercodes oder Länderkennungen	Spezifikation für die Kennzeichnung von Ländernamen.
ISO 4217	Kürzel für Währungen	Liste von Codes für nationale Währungen.
ISO 5218	Kennzeichnungen für das Geschlecht	Codes zur Angabe des Geschlechts.
ISO 6093	Angabe für nummerische Werte	Drei Präsentationsnormen für nummerische Werte. In Form von Zeichenketten (Texten), in einer maschinenlesbaren Form und eine für Menschen gut lesbare Form.
ISO 6709	Ortsangaben	Format zur eindeutigen Identifikation von Ortskoordinaten auf, unter oder über der Erdoberfläche (Längengrad, Breitengrad, Höhe).
ISO 8601	Datum und Zeit	Format für Datums- und Zeitangaben.

Tabelle 6.1: ISO-Standards zur Datenkodierung. Quelle: Anderson u. a. (2000).

6.3.6 Datenverarbeitung

Zur Geodatenverarbeitung wurde im Rahmen der GLASS-Entwicklung eine C++-Klasse implementiert. Diese Klasse bietet Operationen, mit denen grundlegende Transformationen und Kombinationen von rasterbasierten Geodaten durchgeführt werden können. Zu den realisierten Operationen gehören solche zur Multiplikation, Division, Addition und Subtraktion von Rasterdaten mit skalaren Werten sowie mit anderen Rasterdaten. Darüber hinaus bietet die Klasse Operationen zur Berechnung zonaler Summen und zonaler Mittelwerte (s. Bsp. in Abb. 6.17).

Verwendung

Die Klasse zur Geodatenverarbeitung wird sowohl von Teilen des Simulationsmodells eingesetzt als auch von den Werkzeugen, die für die Vorverarbeitung und Nachbearbeitung von Assessment-Daten erstellt wurden. Ein Beispiel zur Verwendung dieser Klasse findet sich im Anhang D.7 (Seite 257).

Utility

Neben der Klasse zur Unterstützung der Geodatenverarbeitung wurden einige allgemeine Datenverarbeitungsfunktionen erstellt; u. a. Funktionen zur Konvertierung der Kodierung von Binärdaten (big/little endian) sowie ein einfacher Zufallszahlengenerator, der für den Klimavariabilitätsgenerator in GLASS benötigt wird.

Raster 1

1	1	2	4
1	2	2	4
1	3	3	4
1	3	3	3

Raster 2

1	1	1	1
1	1	1	1
2	2	2	2
2	2	2	1

Summe

2	2	3	5
2	3	3	5
3	5	5	6
3	5	5	4

zonale Summe

7	7	3	3
7	3	3	3
7	9	9	3
7	9	9	9

Abbildung 6.17: Funktionsbeispiel zur Geodatenverarbeitung. Die Abbildung verdeutlicht die realisierten Operationen zur Bildung von Summen und zonalen Summen von Geodatensätzen. Die Summe ergibt sich aus der Addition der Werte, die sich an der entsprechenden Position von Raster 1 und Raster 2 befinden. Die zonale Summe ergibt sich aus der Summe aller Werte von Raster 2, die sich in *einer* Zone befinden, wobei die Zonen durch die Werte in Raster 1 bestimmt werden.

6.4 Fazit

Ausgangsbasis

In Kapitel 5 (Seite 95) wurde eine Software-Architektur für Systeme zum integrierten simulationsbasierten Assessment (SISA) entwickelt. Diese Architektur soll als Ausgangsbasis für die Entwicklung neuer SISA herangezogen werden können.

Ziel

Ziel dieses Kapitels war es, die *Anwendbarkeit* dieser *allgemeinen* SISA-Architektur anhand einer prototypischen *Implementierung* der spezifizierten Komponenten unter Verwendung eines *konkreten Simulationssystems* zu belegen.

Rahmen

Die Implementierungen erfolgten für die Simulationsmodelle, die im Rahmen des Assessment-Projektes *GLASS (GLobal ASsessment of Security)* – dessen Ziel die Analyse der Auswirklungen des globalen Wandels auf die Wasser- und Nahrungsmittelverfügbarkeit ist – eingesetzt werden. Das GLASS-Simulationssystem besteht sowohl aus Teilmodellen, die bereits in anderen Projekten verwendet wurden (z. B. das Modell WaterGAP) als auch aus Teilmodellen, die speziell für das GLASS-Projekt erstellt wurden (z. B. das Wasserstressmodell und das Nahrungsmittelstressmodell).

Tests

Unter Nutzung des GLASS-Simulationssystems wurden die folgenden zentralen Konzepte der Architektur überprüft:

- die Verwaltung und Bereitstellung wichtiger *Assessment-Informationen* durch die *Dokumentationskomponente*
- die Verwaltung von *Zugriffsinformationen* und *Metadaten* innerhalb des Katalogmanagers
- die automatische Aktualisierung des Katalogmanagers durch *Metadaten-Sammler*
- die *Kapselung* von Simulationsmodellen in der *Simulationssystemkomponente*
- die Bereitstellung von *Simulationslauf-Spezifikationen* über den *Simulationslaufmanager*
- der transparente *Datenzugriff* über die *Datenzugriffskomponente*
- die *Datenspeicherung* im Datenbanksystem bzw. in Dateien (*Datenbasis*)
- die Bereitstellung wieder verwendbarer Funktionen zur *Geodatenverarbeitung* und *allgemeinen Datenverarbeitung*

Ergebnisse

Zusammenfassend lässt sich festhalten, dass die Umsetzung der zentralen Konzepte der entwickelten SISA-Architektur auch mit relativ einfachen technischen Mitteln und unter Verwendung freier Software möglich ist. Bei der Anwendung im Rahmen des GLASS-Modells haben sich insbesondere die Verfolgung der Metadaten-Konzepte (Nutzung der 15 grundlegenden Metadaten-Elemente, Verwendung von eindeutigen Ressourcen-Bezeichner, Nutzung von Metadaten-Sammlern) und die Nutzung des Simulationslaufmanagers als entscheidende Schritte zu mehr Transparenz und Nachvollziehbarkeit erwiesen. Die Übertragung der für das Simulationssystem definierten Schnittstellen (Operationen init, run, getResult) auf die Teilmodelle führt zu einer verbesserten Wiederverwendbarkeit und Interoperabilität der Teilmodelle. Ferner hat sich gezeigt, dass die Informationen der Dokumentationskomponente entscheidend zu einem reibungslosen und transparenten Assessment beitragen können.

Eine Zusammenfassung wichtiger Realisierungsaspekte zu den aufgeführten Tests findet sich in den folgenden Absätzen.

Dokumentation

Zur Verwaltung der Assessment-Informationen (Anmerkungstexte, Informationen über Personen und Organisationen etc.) wurde das relationale Datenbank-Managementsystem (RDBMS) *MySQL* eingesetzt, das die Verwaltung der Datenbestände über einen Web-Browser erlaubt. Zur Bereitstellung und Erweiterung von Datenbeständen wurde ein separates, auf der Programmiersprache *PHP* basierendes, Web-Interface implementiert.

Der Katalogmanager, der die Metadaten und Zugriffsinformationen verwaltet, basiert ebenfalls auf dem RDBMS *MySQL*. Die eindeutige *Identifizierung* von Ressourcen im Katalogmanager wird über *Uniform Resource Names (URN)* realisiert. URNs folgen einer Syntax, die im Rahmen dieses Kapitels speziell für SISA-Ressourcen entwickelt wurde. Die Erstellung eines URN für eine Ressource wird durch einen 'URN-Generator' unterstützt, der über einen Web-Browser bedient werden kann. Jeder SISA-Ressource muss ein solcher URN zugewiesen werden. Nicht abstrakte Ressourcen (vgl. Abb. 4.2, Seite 76) müssen über den URN hinaus einen *Uniform Resource Locator (URL)* besitzen, der über den Ort der Speicherung der Ressource und den Zugriffsmechanismus auf diese Ressource Auskunft gibt. Katalog-manager

Als Metadatensatz wurde der *Dublin Core Metadata Element Set (DCMES)* verwendet. Zur Erfassung der Metadaten wurde ebenfalls eine Web-Seite erstellt. Nach der Eingabe der 15 Elemente sowie des URN und evtl. des URL kann ein automatischer Eintrag in die *Datenbank* des Katalogmanagers erfolgen. Darüber hinaus ist es möglich, die Meatdaten in Form einer *Datei* (im XML/RDF-Format) zu speichern.

Die in Form von *Dateien* gespeicherten Metadaten können über *Metadaten-Sammler* zusammengetragen und in gesonderten Repositories (ebenfalls im XML/RDF-Format) gespeichert werden. Das Programm des Metadaten-Sammlers wurde über ein einfaches *Shell-Skript* realisiert. Die Integration der Repository-Daten in die Katalogkomponente erfolgt durch ein *PHP*-Programm. Dieses Programm verwendet das *Document Object Model (DOM)*, um die Repositories zu analysieren, und überträgt die Metadaten über die *Structured Query Language (SQL)* in die MySQL-Datenbank. Bei dieser Übertragung wird auch automatisch die *Ressourcen-Liste* des Katalogmanagers aktualisiert. Sowohl die Repositories als auch die einzelnen Metadaten-Dateien können durch die Verwendung von *XML-Stylesheets* direkt über einen Web-Browser angezeigt werden. Sammler

Um die Transparenz und Nachvollziehbarkeit von Modellergebnissen zu erhöhen, sollten die Simulationsmodelle bei der Erzeugung von Ergebnisdatensätzen direkt die zugehörigen Metadaten schreiben. Zur Gewährleistung der Nachvollziehbarkeit sollten die Metadaten zu einem Ergebnisdatensatz auf jeden Fall den URN des Modells sowie den URN des Simulationslaufs beinhalten. Um eine derartige Dokumentation von Modellergebnissen zu erleichtern, wurde innerhalb der Komponente zur allgemeinen Datenverarbeitung eine C++-Klasse implementiert, über die die DCMES-Elemente im XML/RDF-Format gespeichert werden können. Metadaten Generierung

Zur Realisierung der Simulationssystem-Komponenten der SISA-Architektur wurde eine C++-Klasse erstellt. Diese Klasse dient als Basisklasse für die im Rahmen des GLASS-Projektes neu erstellten Teilmodelle. In Anlehnung an die für das Simulationssystem der SISA-Architektur definierte Schnittstelle müs- Simulationssystem

sen alle von dieser Basisklasse abgeleiteten Teilmodelle die Operationen *init()* und *run()* implementieren. Des Weiteren muss jedes Teilmodell eine Operation namens *resultsAvailable()* anbieten. Über diese Operation gibt ein Teilmodell Auskunft darüber, ob für eine bestimmte Simulationslaufspezifikation bereits Ergebnisse vorliegen. Die neu erstellten Teilmodelle implementieren außerdem die Schnittstelle *ISimResultAccess* oder bieten eine entsprechende Operation zur Abfrage von Modellergebnissen an (das Nahrungsmittelstressmodell stellt beispielsweise die Operation *getStress()* zur Verfügung).

Das wieder verwendete Modell zur Berechnung von Wasserverfügbarkeiten (WaterGAP) bietet die geforderten Schnittstellen nicht an. Um das Modell dennoch als Teilmodell innerhalb des Simulationssystems benutzen zu können, wurde ein *Adapter* implementiert. Dieser Adapter ist für die Anpassung der Schnittstelle des WaterGAP-Modells an die geforderten Schnittstellen zuständig.

Die neu erstellten Teilmodelle können über einen einfachen, in C++ implementierten *Modellkoppler* miteinander verbunden werden. Um eine solche Verbindung zu erlauben, muss sich ein Modell bei diesem Koppler anmelden. Der Koppler trägt daraufhin einen Verweis auf das Modell in eine interne Liste ein und gibt ihn auf Anfrage an andere Teilmodelle weiter. Die Interaktion der Teilmodelle geschieht dann über deren definierte Schnittstelle.

Simulationslaufmanager

Die für einen Simulationslauf notwendigen Einstellungen (Simulationslaufspezifikationen) erhalten die Simulationsmodelle vom *Simulationslaufmanager*. Der Simulationslaufmanager wurde unter Verwendung der Programmiersprache *C++* als *Server-Anwendung* unter *Microsoft Windows 2000* realisiert. Die Kommunikation zwischen dem Simulationslaufmanager und den Simulationsmodellen (Clients) geschieht über das *Internet-Protokoll (TCP/IP)* (Verwendung von Sockets). Auf diese Weise wird der Aufbau einer verteilten Anwendung ermöglicht (das Modell WaterGAP läuft beispielsweise unter Linux). Die Simulationslaufspezifikationen werden in Form von XML-Dateien gespeichert und vom Simulationslaufmanager eingelesen. Durch die Verwendung einer *Document Type Definition (DTD)* innerhalb der Spezifikationsdatei kann der Simulationslaufmanager die Gültigkeit von Einstellungen über einen validierenden Parser überprüfen.

Datenzugriff

Die Datenzugriffskomponente wurde, genauso wie der Simulationslaufmanager, als *Server-Anwendung* unter *Microsoft Windows 2000* realisiert. Die Simulationsmodelle (Clients) wenden sich, unter Angabe des Datensatz-URN und des gewünschten Formats, in dem der Datensatz geliefert werden soll, an den Server. Der Server wendet sich seinerseits an den Katalogmanager, um die für einen Zugriff notwendigen Informationen (URL und weitere Informationen zum Datenformat) zu erfragen. Anschließend liefert die Datenzugriffskomponente dem Client den Datensatz im gewünschten Format. Die Transformation verschiedener Datenformate geschieht transparent für den Client und wird über

so genannte *Mediatoren* realisiert. Im Rahmen des Architekturtests wurden verschiedene Mediatoren für das Teilmodell WaterGAP realisiert. Die Mediatoren greifen auf die Datensätze aus der Datenbasis zu und geben sie dann im internen (binären) Format (UNF-Format) an das Modell weiter.

Datenbasis

Die Mediatoren erlauben den lesenden und schreibenden Zugriff auf das UNF-Dateiformat und auf das textbasierte Austauschformat für Rasterdaten des GIS ArcView. Darüber hinaus wurden Mediatoren für den Zugriff auf Daten entwickelt, die in einem RDBMS gespeichert sind. Die Verbindung der Mediatoren mit dem RDBMS wurde über die *Open Database Connectivety (ODBC)* realisiert. Durch die ODBC-Verbindung ist ein relativ einfacher Austausch des verwendeten RDBMS möglich (getestet wurden *MySQL* und *Microsoft Access 2000*). Außerdem wurde die etwas schnellere Verbindung zur Datenbank über *Active Data Object (ADO)* implementiert (und für *Microsoft Access 2000* getestet). Weitere Mediatoren erlauben den Export von Datensätzen in das ArcView-Rasterformat, in GeoTiff sowie in Dateien, die die *Geographical Markup Language (GML)* als Format nutzen. Der Mediator zum Export von Daten in das ArcView-Rasterformat ist ein Beispiel für die Kopplung mit einem GIS, da er zur Erzeugung des proprietären Datenformats die Programmierschnittstelle von ArcView nutzt.

Für die Ausgestaltung der Komponenten zur Geodatenverarbeitung bzw. zur allgemeinen Datenverarbeitung wurden einige C++-Klassen und Funktionen implementiert. Die geodatenverarbeitenden Klassen erlauben die Kombination und Transformation von Rasterdaten des internen UNF-Formats. Die Funktionen erlauben z. B. die Addition zweier Rasterdatensätze oder die Bildung zonaler Summen. In der Komponente der allgemeinen Datenverarbeitung befindet sich u. a. ein Zufallszahlengenerator.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Systeme zum integrierten simulationsbasierten Assessment (SISAs) sind Software-Systeme, die von unterschiedlichen Fachdisziplinen stammende Daten und Simulationsmodelle zum ‘System Erde’ in einem konsistenten Rahmen kombinieren und neue Daten über den Zustand und mögliche langfristige Änderungen des ‘Systems Erde’ – vornehmlich zur Unterstützung politischer Entscheidungsträger – berechnen und bereitstellen. Diese Systeme, die von einigen Autoren auch als *integrierte Modelle* bezeichnet werden, sind wichtige Werkzeuge zur Analyse des globalen Wandels. Durch die zunehmende Komplexität und Größe von SISAs ergeben sich Herausforderungen bezüglich der *Transparenz*, *Nachvollziehbarkeit* und *Reproduzierbarkeit* von Analysen und Ergebnissen, der *Erweiterbarkeit* von Modellen, der *Wiederverwendbarkeit* und *Austauschbarkeit* von Modellteilen sowie der *Interoperabilität*. Um diesen Herausforderungen gerecht zu werden, muss das Prinzip der *Modularisierung* angewendet werden. Vorhandene Systeme sind allerdings zumeist unzureichend modularisiert (Jaeger u. a., 2002) und genügen daher nicht den gestiegenen Anforderungen. Problem

Die Modularisierung kann über die Definition einer *Software-Architektur* erreicht werden. Eine Software-Architektur ist die *grundsätzliche Strukturierung* eines Software-Systems. Sie beschreibt eine Menge definierter *Komponenten*, die über *Schnittstellen* miteinander kommunizieren, spezifiziert deren jeweiligen *Zuständigkeitsbereich* und beschreibt die *Beziehungen* zwischen den Komponenten. Lösungsansatz

Wie eine Analyse existierender Systeme in Kapitel 3 (Seite 25) gezeigt hat, spiegeln sich die gestiegenen Leistungsanforderungen an die Modelle (z. B. die

Integration von GIS-Funktionalitäten) zwar in den einzelnen Systemen wider, eine *Komponentenbildung* im Sinne einer Software-Architektur konnte in der Literatur aber *nicht* gefunden werden. Die Systeme werden hingegen zumeist in Module eingeteilt, die sich aus der *Realisierung* der Systeme ergeben (z. B. in Klassen-Bibliotheken). Aufgrund der unterschiedlichen Funktionalitäten der Module, lassen sich diese nicht ohne weiteres unter den Systemen austauschen. Eine Interoperabilität zwischen den Systemen ist wegen der uneinheitlichen Einteilung der Gesamtsysteme sowie der unterschiedlichen Implementierungsmethoden bei der Funktionsrealisierung ebenfalls nicht gegeben.

Ziel

Das Ziel der vorliegenden Arbeit war daher die Entwicklung einer allgemein anwendbaren *Software-Architektur* für SISAs, die die *Wiederbenutzbarkeit* und *Wiederbenutzung* von Modellen, Modellteilen, Daten und anderen notwendigen *Betriebsmitteln* unterstützt, die *Zusammenarbeit* mit anderen Programmen begünstigt und die *Qualität* der Ergebnisse sichern hilft.

Fragen

Um dieses Ziel zu erreichen, stellten sich die folgenden Forschungsfragen:

- In welche generellen Komponenten sollte ein System zum integrierten simulationsbasierten Assessment aufgeteilt werden?
- Welche Komponenten können unabhängig von *einem* konkreten System *realisiert* und damit für unterschiedliche Modelle wieder verwendet werden?
- Welche Daten sollten zur Unterstützung der Transparenz von Analyse- und Simulationsergebnissen vorgehalten werden?
- Welche Standards können zur Erhöhung der Qualität integrierter Modelle beitragen?

Systemdefinition

Der generelle Ausgangspunkt für die Entwicklung einer Software-Architektur ist die *Systemdefinition*, in der die Hauptfunktionen und Hauptdaten sowie die grundlegenden Anforderungen an ein System bestimmt werden. Um die allgemeinen, auf mehrere Systeme übertragbaren Anforderungen eines SISA zu definieren, wurde in Kapitel 3 (Stand der Technik, Seite 25) die Literatur über SISAs hinsichtlich ihrer Leistungsmerkmale und ihres Funktionsumfangs ausgewertet. Diese Analyse zeigte, dass sich das Leistungsspektrum von SISAs nicht auf die Berechnung neuer Simulationsergebnisse beschränkt: insbesondere die Bereitstellung und Nutzung von GIS-Funktionalitäten, die Unterstützung bei der Analyse von Ergebnissen, Daten und Modellen, die Verwaltung von Szenarien und Datenbeständen sowie die Bereitstellung von Systemdokumentationen gehören ebenfalls zu den Leistungsmerkmalen vieler Systeme. Aufbauend auf diesem Ergebnis und einer weiteren Analyse nicht-funktionaler (qualitativer) Anforderungen wurde in Kapitel 4 (Seite 69) eine *Systemdefinition* für ein allgemeines (d. h. nicht von konkreten Projektanforderungen beeinflusstes) SISA vorgenommen. Der folgende Absatz fasst die wichtigsten Aspekte der Systemdefinition zusammen.

Leistungsspektrum

Systemziel

Ressourcen

Daten

Simulationen

Konsistenz

Umgebung

Qualität

Ziel des Systems zum integrierten simulationsbasierten Assessment ist die Unterstützung des integrierten Assessments durch die Bereitstellung eines konsistenten Rahmens für Daten und Simulationsmodelle zum System Erde und zur Durchführung von Simulationsläufen sowie die Bereitstellung grundlegender Informationen zu durchgeführten oder in der Durchführung begriffenen Projekten. Um dieses Systemziel zu erreichen, müssen verschiedene Betriebsmittel (Ressourcen) durch das SISA verwaltet werden. Zu diesen Ressourcen gehören sowohl die Simulationsmodelle und die ihnen zugeordneten Daten als auch andere Software (z. B. zur Vorverarbeitung oder Nachbearbeitung von Daten) und Dokumente (z. B. Modellbeschreibungen oder Ergebnisberichte). Darüber hinaus muss das SISA Informationen über Projekte, Analysen, Szenarien, beteiligte Personen und andere Hintergrundinformationen bereitstellen. Die Daten, die für die Simulationsmodelle benötigt werden, sollten ebenfalls über das SISA zur Verfügung stehen. Die Durchführung und Verwaltung von Simulationsläufen gehört darüber hinaus ebenso zur Aufgabe des SISA wie die Bereitstellung von Simulationsergebnissen. Die Sicherstellung der Konsistenz wird unterstützt durch die Dokumentation der Simulationsergebnisse, der verwendeten Simulationsmodelle, der zugrunde liegenden Simulationslauf-Spezifikation und des Simulationslaufes selbst. Das SISA sollte in eine Software-Umgebung eingebettet werden können. Insbesondere sollten Schnittstellen zu Geo-Informationssystemen (GIS) und zu Datenbank-Managementsystemen (DBMS) vorhanden sein. Neben der gewünschten *Interoperabilität* mit GIS und DBMS stellt das SISA weitere Anforderungen an die Qualität der Software-Architektur (nicht-funktionale Anforderungen). Wegen des zunehmend notwendigen Austausches von Modellteilen zwischen unterschiedlichen Organisationen sollte das Qualitätsmerkmal der *Austauschbarkeit* (zumindest von Modellteilen) bei der Entwicklung der Architektur besonders berücksichtigt werden. Die *Modifizierbarkeit* des Systems (insbesondere von Modellteilen) sollte wegen der oft notwendigen Änderungen und Aktualisierungen ebenfalls sehr gut sein. Darüber hinaus wird die *Transparenz, Nachvollziehbarkeit und Reproduzierbarkeit* von Assessment-Ergebnissen als wichtiges Qualitäts-Merkmal eines SISA definiert. Um die Anforderung der *Nachvollziehbarkeit* von Assessment-Ergebnissen zu erfüllen, ist das Qualitätsmerkmal der *Analysierbarkeit* ebenfalls besonders zu berücksichtigen.

Architektur

Diese allgemeine Systemdefinition wurde in Kapitel 5 (Seite 95) als Basis für die Entwicklung der *Software-Architektur* für ein SISA benutzt. Hinweise zur Abgrenzung von Komponenten wurden der bereits angesprochenen Analyse vorhandener SISAs entnommen. Da die Wiederbenutzbarkeit und Interoperabilität im Zusammenhang mit SISAs eine wichtige Rolle spielt, wurden zur Abgrenzung und Definition der Architektur-Komponenten auch *Standards* berücksichtigt. Wichtige Standards, die bei der Entwicklung der Software-Architektur von Interesse sind, wurden in Kapitel 3 (Seite 25) vorgestellt. Ein SISA ist

ein geodatenverarbeitendes System. Aus diesem Grund sind bei dessen Entwicklung insbesondere die Arbeiten des *technischen Komitees für geographische Informationen/Geomatik (TC211)* der *internationalen Organisation für Standardisierung (ISO)* sowie die Arbeiten des *Open-GIS-Konsortiums (OGC)* von Bedeutung. Den gemeinsamen *Rahmen* für die Standards des TC211 und des OGC bildet der *ISO/DIS 19119*. Dieser Standard, der die Empfehlung eines grundsätzlichen architektonischen Aufbaus für geodatenverarbeitende Systeme enthält, wurde sowohl zur Komponentenabgrenzung als auch für die Schnittstellendefinitionen der SISA-Architektur herangezogen.

12 Komponenten

Um die definierten Systemziele zu erfüllen, wurde die Architektur des SISA in *zwölf* Komponenten geteilt. Die zentrale Komponente der Architektur ist die *Simulationssystem-Komponente*, die für die Berechnung, Speicherung und

Simulationssystem

Weitergabe von Simulationsergebnissen verantwortlich ist. Die Weitergabe von Simulationsergebnissen erfolgt über eine gesonderte Schnittstelle. Die Möglichkeit Ergebnisdaten direkt vom Simulationssystem abfragen zu können erhöht die Interoperabilität und Wiederverwendbarkeit des Simulationssystems. Um

Simulationslauf-Manager

die Nachvollziehbarkeit und Reproduzierbarkeit von Simulationsergebnissen zu gewährleisten, werden alle für einen Simulationslauf benötigten Einstellungen von einer separaten Komponente verwaltet und bereitgestellt: dem *Simulationslaufmanager*. Die Referenzierung von Datensätzen wird nicht über den Ort der Datenspeicherung (z. B. einem Dateinamen), sondern über eindeutige

Datenzugriff

Ressourcen-Namen vorgenommen. Das Simulationssystem greift auf einen derart referenzierten Datensatz nicht direkt zu. Stattdessen wendet es sich an die *Datenzugriffskomponente*, die für den lesenden und schreibenden Zugriff auf Daten und die Transformation zwischen verschiedenen Daten-Formaten zuständig ist. Diese Art des Datenzugriffs ermöglicht einen ortstransparenten Datenzugriff sowie eine automatische Transformationen von Datenformaten. Das Format der Datenspeicherung wird somit von der 'internen' Repräsentation für das Simulationssystem getrennt. Dieser Zugriffsmechanismus erlaubt eine schrittweise Mi-

Katalogmanager

gration hin zu offenen Datenformaten. Die Informationen, die die Datenzugriffskomponente zum Zugriff auf einen Datensatz benötigt (z. B. den Dateinamen), werden vom *Katalogmanager*, der für die Verwaltung von *Metadaten* zuständig ist, bereitgestellt. Die Beschreibung jeder Ressource über die 15 Elemente des *Dublin Core Metadata Element Set* (ISO 15836) wird als Minimal-Anforderung

Datenbanksystem

der Dokumentation angesehen. Die Datenzugriffskomponente greift auf die Daten innerhalb der Datenbasis zu. Die Datenbasis besteht aus einer *Datenbanksystem-Komponente*, die für die verwaltete Speicherung von Assessment-Daten zuständig ist, und kann durch eine lose Sammlung von Dateien ergänzt werden. Die Integration der Dateien in das SISA erfolgt über Metadaten, die zu

Metadaten-Sammler

jeder Datei vorhanden sein sollten. Die in der Datenbasis vorhandenen Metadaten, die nicht direkt vom Nutzer über den Katalogmanager eingegeben werden, werden von *Metadaten-Sammlern* verarbeitet. Ein Metadaten-Sammler ist ver-

antwortlich für die Durchsuchung eines Rechners nach Dateien mit Metadaten und die automatische Weitergabe der gefundenen Informationen an den Katalogmanager. Auf diese Weise wird die Wiederverwendbarkeit von Ressourcen erhöht. Ein Metadaten-Sammler sollte auf jedem Rechner installiert sein, der Ressourcen für das SISA bereitstellt.

Projektbezogene Kurz-Informationen werden direkt im SISA vorgehalten, genauer: in der *Dokumentationskomponente*. Diese Komponente ist verantwortlich für die Dokumentation und Verwaltung wichtiger Assessment-Informationen. In diesem 'Auskunftssystem' werden u. a. Daten über durchgeführte Simulationsläufe, beteiligte Personen und verwendete Szenarien hinterlegt und den Akteuren des SISA (Modellbetreiber, Modellentwickler, Entscheidungsträger, Interessenten) bereitgestellt. Die Daten dieser Komponenten tragen entscheidend zur Transparenz von Assessment-Ergebnissen bei. Dokumentation

Zur Steigerung der Wiederverwendbarkeit von Software sollten häufig wiederkehrende Funktionen nicht direkt in der Komponente des Simulationssystems implementiert, sondern in andere Komponenten ausgelagert werden. Im Rahmen des SISA wurden drei thematische Komponenten abgegrenzt: Die *Geodatenverarbeitungs-Komponente*, die für die Verarbeitung geographischer Daten und die Bereitstellung einer Schnittstelle zu eigenständigen GIS verantwortlich ist, die Komponente zur *allgemeinen Datenverarbeitung*, die zuständig ist für die Bereitstellung allgemeiner, wieder verwendbarer Datenverarbeitungsdienste, und die Komponente der *Aufgabensteuerung*, in deren Verantwortung der programmgesteuerte Aufruf anderer Funktionen des SISA liegt (z. B. Funktionen zur Datenvorverarbeitung und Datennachbearbeitung). Datenverarbeitung

Zur Sensitivitäts- und Unsicherheitsanalyse von Simulationsmodellen ist eine gesonderte Komponente vorgesehen: die *Modellanalyse-Komponente*. Diese Komponente schaltet sich zur Modellanalyse als Schicht zwischen das Simulationssystem und die Datenzugriffskomponente. Auf diese Weise kann die Komponente die Eingabedaten für das Simulationsmodell gezielt verändern und die Ausgabedaten analysieren. Zur Modellanalyse müssen daher keine Änderungen innerhalb des Simulationssystems vorgenommen werden. Funktionen, die speziell für konkrete Assessments (Projekte) benötigt werden, sollen zur Steigerung der Transparenz, Analysierbarkeit und Modifizierbarkeit des System der Komponente *Ergebnisanalyse* zugeordnet werden. Modell-Analyse Ergebnisanalyse

Die Simulationssystemkomponente sowie die Komponente zur Ergebnisanalyse sind die einzigen Komponenten, die für ein neues SISA angepasst werden müssen. Die anderen Komponenten können generell für SISAs anderer Projekte wieder verwendet werden (von Erweiterungen um neue Funktionen, Datentypen etc. abgesehen). Wiederverwendbarkeit

Um die *Anwendbarkeit* der *allgemeinen* SISA-Architektur zu belegen, wurden die zentralen Komponenten unter Verwendung eines *konkreten Simulationssystems* prototypisch *implementiert* (Kapitel 6, Seite 165). Die Implementie- Realisierung

rungen erfolgten für die Simulationsmodelle, die im Rahmen des Assessment-Projektes *GLASS (GLobal ASsessment of Security)* eingesetzt wurden. Ziel des Projektes ist die Analyse der Auswirkungen des globalen Wandels auf die Wasser- und Nahrungsmittelverfügbarkeit. Das GLASS-Simulationssystem besteht sowohl aus Teilmodellen, die bereits in anderen Projekten verwendet wurden (z. B. das Modell WaterGAP) als auch aus Teilmodellen, die speziell für das GLASS-Projekt erstellt wurden (z. B. das Wasserstressmodell und das Nahrungsmittelstressmodell).

Tests

Unter Nutzung des GLASS-Simulationssystems wurden die folgenden zentralen Konzepte der Architektur überprüft:

- die Verwaltung und Bereitstellung wichtiger *Assessment-Informationen* durch die *Dokumentationskomponente*
- die Verwaltung von *Zugriffsinformationen* und *Metadaten* innerhalb des Katalogmanagers
- die automatische Aktualisierung des Katalogmanagers durch *Metadaten-Sammler*
- die *Kapselung* von Simulationsmodellen in der *Simulationssystemkomponente*
- die Bereitstellung von *Simulationslauf-Spezifikationen* über den *Simulationslaufmanager*
- der transparente *Datenzugriff* über die *Datenzugriffskomponente*
- die *Datenspeicherung* im Datenbanksystem bzw. in Dateien (*Datenbasis*)
- die Bereitstellung wieder verwendbarer Funktionen zur *Geodatenverarbeitung* und *allgemeinen Datenverarbeitung*

Ergebnisse

Durch die prototypische Implementierung konnte gezeigt werden, dass die Umsetzung der zentralen Konzepte der entwickelten SISA-Architektur auch mit relativ einfachen technischen Mitteln und unter Verwendung freier Software möglich ist. Bei der Anwendung im Rahmen des GLASS-Modells haben sich insbesondere die Verfolgung der Metadaten-Konzepte (Nutzung der 15 grundlegenden Metadaten-Elemente, Verwendung von eindeutigen Ressourcen-Bezeichnern, Nutzung von Metadaten-Sammlern) und die Nutzung des Simulationslaufmanagers als entscheidende Schritte zu mehr Transparenz und Nachvollziehbarkeit erwiesen. Die Übertragung der für das Simulationssystem definierten Schnittstellen (Operationen init, run, getResult) auf die Teilmodelle führt zu einer verbesserten Wiederverwendbarkeit und Interoperabilität der Teilmodelle. Ferner hat sich gezeigt, dass die Informationen der Dokumentationskomponente entscheidend zu einem reibungslosen und transparenten Assessment beitragen können.

7.2 Ausblick

Die Anwendbarkeit der grundlegenden Prinzipien der entwickelten SISA-Architektur konnte anhand einer prototypischen Implementierung gezeigt werden. Der nächste Schritt wäre die Entwicklung eines SISA, das vollends auf den beschriebenen Komponenten beruht. Hierzu müssten zunächst schnittstellenkonforme Realisierungen der wieder verwendbaren Komponenten implementiert und getestet werden. Bei der Realisierung der Komponenten sollte die Verwendung von Web-Technologien in Betracht gezogen werden, da diese einen entscheidenden Schritt in Richtung Interoperabilität versprechen. Eine weitere Verbesserung der Architektur kann durch die Integration von Funktionen zur Metadaten-Speicherung in die Datenzugriffskomponente erfolgen. Eine solche Integration würde die obligatorische Angabe von Metadaten für alle zu speichernden Datensätze erlauben und somit die Qualitätsmerkmale der Nachvollziehbarkeit und Wiederverwendbarkeit, insbesondere von Simulationsergebnissen, verbessern. Ein Abgleich der Schnittstellen der Datenzugriffskomponente mit weiteren Spezifikationen des OpenGIS-Konsortiums verspricht ebenfalls eine Verbesserung der Interoperabilität eines SISA und sollte aus diesem Grund vorgenommen werden.

Literaturverzeichnis

[ADEPT 2001] Alexandria Digital Earth Prototype – Metadata for Models Working Group (Hrsg.): *Content Standard for Computational Models*. Version 1.2. Santa Barbara, USA : University of California, 2001. – URL http://www.ncgia.ucsb.edu/projects/metadata/standard/standard_1.2.doc

[Alcamo 1994] Alcamo, Joseph (Hrsg.): *IMAGE 2.0: Integrated modelling of global climate change*. Dordrecht, Boston, London : Kluwer Academic Publishers, 1994. – reprinted from Water, Air, and Soil Pollution, Volume 76, Nos. 1-2, 1994. – ISBN 0-7923-2860-4

[Alcamo 2001] Alcamo, Joseph: *Scenarios as tools for international environmental assessment*. Luxembourg : European Environmental Agency, Office for Official Publication of the European Communities, 2001 (Experts' corner report, Prospects and scenarios No. 5; Environmental issues report No. 24). – ISBN 92-9167-402-8

[Alcamo 2002] Alcamo, Joseph: Three issues for improving integrated models: uncertainty, social science, and legitimacy. In: Gethmann, Carl F. (Hrsg.) ; Lingner, Stephan (Hrsg.): *Integrative Modellierung zum Globalen Wandel*. Berlin, Heidelberg, New York u.a. : Springer, 2002 (Wissenschaftsethik und Technikfolgenbeurteilung, Band 17), S. 3–14. – ISBN 3-540-43253-1

[Alcamo u. a. 2003a] Alcamo, Joseph ; Döll, Petra ; Henrichs, Thomas ; Kaspar, Frank ; Lehner, Bernhard ; Siebert, Stefan: Development and testing of the WaterGAP 2 global model of water use and availability. In: *Hydrological Sciences Journal* 48 (2003), Nr. 3, S. 317–338

[Alcamo u. a. 2003b] Alcamo, Joseph ; Döll, Petra ; Henrichs, Thomas ; Kaspar, Frank ; Lehner, Bernhard ; Siebert, Stefan: Global estimates of water withdrawals and availability under current and future 'business-as-usual' conditions. In: *Hydrological Sciences Journal* 48 (2003), Nr. 3, S. 339–348

[Alcamo u. a. 2003c] Alcamo, Joseph ; Dronin, Nikolai ; Endejan, Marcel ; Golubev, Genady ; Kirilenko, Andrei: *Will Climate Change Affect Food and Water Security in Russia? Summary Report of the International Project on Global Environmental Change and its Thread to Food and Water Security in Russia.* Kassel : Center for Environmental Systems Research, Universtity of Kassel, 2003 (Report No. A0302). – URL http://www.usf.uni-kassel.de/usf/archiv/dokumente/projekte/rglass.summary.pdf

[Alcamo u. a. 2001] Alcamo, Joseph ; Endejan, Marcel ; Kaspar, Frank ; Rösch, Thomas: The GLASS model: a strategy for quantifying global environmental security. In: *Environmental Science and Policy* 4 (2001), Nr. 1, S. 1–12

[Alcamo u. a. 1998a] Alcamo, Joseph ; Kreileman, Eric ; Krol, Maarten ; Leemans, Rik ; Bollen, Johannes ; Minnen, Jelle van ; Schaeffer, Michiel ; Toet, Sander ; Vries, Bert de: Global modelling of environmental change: an overview of IMAGE 2.1. In: (Alcamo u. a., 1998b), S. 3–94

[Alcamo u. a. 1998b] Alcamo, Joseph (Hrsg.) ; Leemans, Rik (Hrsg.) ; Kreileman, Eric (Hrsg.): *Global Change Scenarios of the 21st Century - Results from the IMAGE 2.1 Model.* Oxford, UK : Pergamon, 1998

[Alcamo u. a. 1990] Alcamo, Joseph (Hrsg.) ; Shaw, Roderick (Hrsg.) ; Hordijk, Leen (Hrsg.): *The Rains Model of Acidification – Science and Strategies in Europe.* Dordrecht; Boston; London : Kluwer Academic Publishers, 1990. – ISBN 0-7923-0781-X (HB), 0-7923-0782-8 (PB)

[Anderson u. a. 2000] Anderson, Richard ; Birbeck, Mark ; Kay, Michael ; u.a.: *XML Professionell.* 1. Aufl. Bonn : MITP-Verlag, 2000. – 957 Seiten

[ANZLIC 2001] Group, Australia New Zealand Land Information Council Metadata W. (Hrsg.): *ANZLIC Metadata Guidelines: Core metadata elements for geographic data in Autralia and New Zealand.* Version 2. Belconnen, Australia : ANZLIC, 2001. – URL http://www.anzlic.org.au/asdi/metagrp.htm

[Bakkes u. a. 2000] Bakkes, Jan A. ; Grosskurth, Jasper ; Idenburg, Annemarth M. ; Rothman, Dale .. ; Vuuren, Detlef P. van: *Description of selected global models for scenario studies on environmentally sustainable development.* Bilthoven, The Netherlands : National Institut of Public Health and the Environment (RIVM), 2000 (Global Dynamics and Sustainable Development Programme, Global Report Series No. 30). – URL http://www.rivm.nl/bibliotheek/rapporten/402001018.pdf. – RIVM Report No. 402001018

[Balzert 1996] Balzert, Helmut: *Lehrbuch der Software-Technik: Software-Entwicklung*. Heidelberg, Berlin, Oxford : Spektrum Akademischer Verlag, 1996 (Lehrbücher der Informatik). – ISBN 3-8274-0042-2

[Balzert 1998] Balzert, Helmut: *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Heidelberg, Berlin : Spektrum Akademischer Verlag, 1998 (Lehrbücher der Informatik). – ISBN 3-8274-0065-1

[Balzert 2000] Balzert, Helmut: *Lehrbücher der Informatik*. Bd. 1. Software-Entwicklung: *Lehrbuch der Software-Technik*. 2. Aufl. Heidelberg, Berlin : Spektrum, Akad. Verlag, 2000. – ISBN 3-8274-0480-0

[Bartelme 2000] Bartelme, Norbert: *Geoinformatik: Modelle, Strukturen, Funktionen*. 3., erweiterte u. aktualisierte Aufl. Berlin, Heidelberg, New York : Springer-Verlag, 2000. – ISBN 3-540-65988-9

[Benz u. a. 1997] Benz, Joachim ; Hoch, Ralf ; Gabele, Tobias: Documentation of Mathematical Models in Ecology – an Unpopular Task? In: *International Society for Ecological Modelling's (ISEM) Newsletter (Ecomod)* (1997), December, S. 1–7

[Benz u. a. 2001] Benz, Joachim ; Hoch, Ralf ; Legovic, Tarzan: ECOBAS – modelling and documentation. In: *Ecological Modelling* (2001), Nr. 1–3, S. 3–15

[Berners-Lee u. a. 1998] Berners-Lee, Tim (Hrsg.) ; Fielding, Roy T. (Hrsg.) ; Masinter, Larry (Hrsg.): *Uniform Resource Identifiers (URI): Generic Syntax*. IETF, 1998 (RFC 2396). – URL http://www.ietf.org/rfc/rfc2396.txt

[Berners-Lee u. a. 1994] Berners-Lee, Tim (Hrsg.) ; Masinter, Larry (Hrsg.) ; McCahill, Mark (Hrsg.): *Uniform Resource Locators (URL)*. CERN, 1994 (RFC 1738). – URL http://www.ietf.org/rfc/rfc1738.txt?number=1738

[Bill und Fritsch 1994] Bill, Ralf ; Fritsch, Dieter: *Grundlagen der Geo-Informationssysteme*. Bd. 1 (Hardware, Software und Daten). 2. Auflage. Heidelberg : Wichmann, 1994. – ISBN 3-87907-265-5

[Boosch u. a. 1999] Boosch, Grady ; Rumbaugh, James ; Jacobson, Ivar: *Das UML-Benutzerhandbuch*. 2. Aufl. Bonn : Addison Wesley, 1999 (Professionelle Softwareentwicklung). – ISBN 3-8273-1486-0

[Bossel 1994] Bossel, Hartmut: *Modellbildung und Simulation: Konzepte, Verfahren und Modelle zum Verhalten dynamischer Systeme*. 2. veränderte Auflage. Braunschweig; Wiesbaden : Vieweg, 1994

[Bratley u. a. 1987] Bratley, Paul ; Fox, Bennett L. ; Schrage, Linus E.: *A Guide to Simulation*. Second Edition. Berlin; Heidelberg; New York : Springer, 1987. – ISBN 3-540-96467-3 u. 0-387-96467-3

[Busch u. a. 2002] Busch, C. ; David, O. ; Kralisch, S. ; Krause, P.: Using the Object Modelling System OMS for future proof hydrological model development and application. In: *Evironmental Modelling and Software (submitted)* (2002)

[Buschmann u. a. 1998] Buschmann, Frank ; Meunier, Regine ; Rohnert, Hans ; Sommerlad, Peter ; Stal, Michael: *Pattern-orientierte Software-Architektur: Ein Pattern-System*. Bonn, Paris u.a. : Addison-Wesley-Longman, 1998. – ISBN 3-8273-1282-5

[Carson 2000] Carson, George S.: Spatial standardization. In: *ACM SIGGRAPH Computer Graphics* 34 (2000), Nr. 3, S. 38–41. – ISSN 0097-8930

[Chen und Norman 1992] Chen, Minder ; Norman, Ronald J.: A Framework for Integrated CASE. In: *IEEE Software* 9 (1992), March/April, Nr. 2, S. 18–22

[Clark u. a. 1975] Clark, John ; Cole, Sam ; Curnow, Ray ; Hopkins, Mike: *Global Simulation Models – A Comparative Study*. London, New York, Sydney, Toronto : John Wiley & Sons, 1975. – ISBN 0-471-15899-2

[Cocks u. a. 1998] Cocks, A.T. ; Rodgers, I.R. ; Skeffington, R.A. ; Webb, A. H.: The limitations of integrated assessment modelling in developing air pollution cotrol policies. In: *Environmental Pollution* 102 (1998), Nr. S1, S. 635–639

[Cook und Daniels 1994] Cook, Steve ; Daniels, John: *Designing Object Systems - Object-Oriented Modelling with Syntropy*. New York, London, Toronto, Sydney, Tokyo, Singopore : Prentice Hall, 1994 (Object-Oriented Series)

[Cox u. a. 2003] Cox, Simon (Hrsg.) ; Daisey, Paul (Hrsg.) ; Lake, Ron (Hrsg.) ; Portele, Clemens (Hrsg.) ; Whiteside, Arliss (Hrsg.): *OpenGIS® Geography Markup Language (GML) Implementation Specification*. Version 3.0. Wayland, Massachusetts, USA : Open GIS Consortium, 2003. – URL http://www.opengis.org/techno/documents/02-023r4.pdf. – Project document number: OGC 02-023r4

[Daigle u. a. 2002] Daigle, Leslie L. ; Gulik, Dirk-Willem van ; Iannella, Renato ; Faltstrom, Patrik: *Uniform Resource Names (URN) Namespace Definition Mechanisms*. IETF, October 2002 (RFC 3406). – URL http://www.ietf.org/rfc/rfc3406.txt?number=3406

[de Bruin 1996] de Bruin, Jos: *Getting Started with M*. Bilthoven, Netherland: National Institute of Public Health and the Environment (RIVM) (Veranst.), 1996. – URL www.m.rivm.nl/html/start/start.htm

[de Bruin u. a. 1996] de Bruin, Jos ; de Vink, Pascal ; van Wijk, Jarke: M – A Visual Simulation Tool. In: *Simulation in the Medical Sciences Conference, Proceedings of the 1996 Western Multiconference*. San Diego : The Society for Computer Simulation, 1996, S. 181–186. – URL www.m.rivm.nl

[DIN 1994] Deutsches Institut für Normung e.V. (Hrsg.): *DIN 66272: Bewertung von Softwareprodukten – Qualitätsmerkmale und Leitfaden zu ihrer Verwendung*. Berlin, Wien, Zürich : Beuth, Oktober 1994 (Informationstechnik). – (Identisch mit ISO/IEC 9126:1991)

[DIN 1995] Deutsches Institut für Normung e.V. (Hrsg.): *DIN-Taschenbuch 166: Software – Enticklung, Dokumentation, Qualität. Normen (Informationstechnik 4)*. 4. Aufl., Stand der abgedr. Normen: August 1995. Berlin, Wien, Zürich : Beuth, 1995. – ISBN 3410134522

[DIN 2000] Deutsches Institut für Normung e.V. (Hrsg.): *DIN EN ISO 9000 Qualitätsmanagementsysteme – Grundlagen und Begriffe*. Berlin, Wien, Zürich : Beuth, Dezember 2000

[Döll u. a. 2003] Döll, Petra ; Kaspar, Frank ; Lehner, Bernhard: A global hydrological model for deriving water availability indicators: model tuning and validation. In: *Journal of Hydrology* 270 (2003), Nr. 1–2, S. 105–134

[Dowlatabadi 1995] Dowlatabadi, Hadi: Integrated assessment models of climate change – An incomplete overview. In: *Energy Policy* 23 (1995), Nr. 4/5, S. 289–296

[Duden 1996] Drosdowski, Günther (Hrsg.) ; Müller, Wolfgang (Hrsg.) ; Scholze-stubenrecht, Werner (Hrsg.) ; Wermke, Matthias (Hrsg.): *Duden – Deutsches Universalwörterbuch*. 3., neu bearbeitete und erweiterte Auflage. Mannheim, Leipzig, Wien, Zürich : Dudenverlag, 1996

[Easterling 1997] Easterling, William E.: Why regional studies are needed in the development of full-scale integrated assessment modelling of global change processes. In: *Global Environmental Change* 7 (1997), Nr. 4, S. 337–356

[Engesser 1993] Lektorat des B.I.-Wissenschaftsverlags unter Leitung von Hermann Engesser (Hrsg.): *Duden 'Informatik': ein Sachlexikon für Studium und Praxis*. 2., vollst. überarb. und erw. Aufl. Mannheim; Leipzig; Wien; Zürich : Dudenverlag, 1993

[Farooqui u. a. 1995] Farooqui, Kazi ; Logrippo, Luigi ; Meer, Jan de: The ISO Reference Model for Open Distributed Processing: an introduction. In: *Computer Networks and ISDN Systems* 2 (1995), Nr. 8, S. 1215–1229

[FGDC 1998] Federal Geographic Data Committee (Hrsg.): *Content Standard for Digital Geospatial Metadata*. Version 1998. Federal Geographic Data Committee, 1998. – URL http://www.fgdc.gov/standards/documents/standards/metadata/v2_0698.pdf. – FGDC-STD-001-1998

[Fink 2002] Fink, Alexander: Szenariotechniken. In: Sommerlatte, Tom (Hrsg.): *Angewandte Systemforschung – Ein interdisziplinärer Ansatz*. Wiesbaden : Gabler, 2002, Kap. 3.4, S. 297–319. – ISBN 3-409-11879-9

[Fischer u. a. 2000] Fischer, Günther ; Velthuizen, Harrij van ; Nachtergaele, Freddy O.: *Global Agro-Ecological Zones Assessment: Methodology and Results*. International Institute for Applied Systems Analysis, 2000 (Interim Report IR-00-064). – URL http://www.iiasa.ac.at/Publications/Documents/IR-00-064.pdf

[Fitzke und Müller 2000] Fitzke, Jens ; Müller, Markus: Simple Features in der Praxis: OpenGIS-Strukturen in Auskunftssystemen für Umwelt- und Naturschutz. In: Cremers, Armin B. (Hrsg.) ; Greve, Klaus (Hrsg.): *Umweltinformatik '00, Umweltinformation für Planung, Politik und Öffentlichkeit. 14. Internationales Symposium „Informatik für den Umweltschutz" der Gesellschaft für Informatik (GI), Bonn 2000* Bd. 1. Marburg : Metropolis Verlag, 2000, S. 484–492

[Foegen und Battenfeld 2001] Foegen, Malte ; Battenfeld, Jörg: Die Rolle der Architektur in der Anwendungsentwicklung. In: *Informatik-Spektrum* 24 (2001), Nr. 5, S. 290–301

[Gamma u. a. 1996] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Entwurtsmuster: Elemente wiederverwendbarer objektorientierter Software*. München; Boston; San Francisco u.a. : Addison-Wesley, 1996 (Professionelle Softwareentwicklung). – ISBN 3-89319-950-0

[Goldfarb und Prescod 2000] Goldfarb, Charles F. ; Prescod, Paul: *The XML Handbook*. 2nd Edition. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2000. – ISBN 0-13-014714-1

[Gordon 2000] Gordon, Alan: *The COM and COM+ programming primer*. Upper Saddle River, New Jersey, USA : Prentice Hall PTR, 2000 (Prentice Hall PTR Microsoft Technologies Series). – ISBN 0-13-085032-2

[Grams 1992] Grams, Timm: *Simulation: strukturiert und objektorientiert programmiert*. Mannheim; Leipzig;Wien; Zürich : BI Wissenschaftsverlag, 1992. – ISBN 3-411-15631-7

[Greenspan und Bulger 2001] GREENSPAN, Jay ; BULGER, Brad: *MySQL/PHP-Datenbankanwendungen*. Bonn : mitp-Verlag, 2001. – ISBN 3-8266-0805-4

[Gulbins und Obermayr 1995] GULBINS, Jürgen ; OBERMAYR, Karl: *Unix System V.4: Begriffe, Konzepte, Kommandos, Schnittstellen*. 4., überarb. Aufl. Bonn, Rending u.a. : Springer Compass, 1995. – ISBN 3-540-58864-7

[Hennicker u. a. 2003] HENNICKER, Rolf ; BARTH, Michael ; KRAUS, Andreas ; LUDWIG, Matthias: An Integrated Simulation System for Global change Research in the Upper Danube Basin. In: *First International NAISO Symposium on Information Technologies in Environmental Engineering (ITEE), June 24 - June 27, 2003* Gdansk University of Technology, Poland (Veranst.), URL http://www.glowa-danube.de/PDF/Publications/100041-00-RH-064.pdf, 2003

[Hering u. a. 2000] HERING, Ekbert ; GUTEKUNST, Jürgen ; DYLLONG, Ulrich: *Handbuch der praktischen und technischen Informatik*. 2., neubearbeitete und erweiterte Auflage. Berlin, Heidelberg, New York : Springer, 2000. – ISBN 3-540-67626-0

[Herold 1999a] HEROLD, Helmut: *Linux-Unix-Kurzreferenz*. 2., überarb. Aufl. Bonn; Reading, Mass. u.a. : Addison-Wesley-Longman, 1999 (Linux/Unix und seine Werkzeuge). – ISBN 3-8273-1536-0

[Herold 1999b] HEROLD, Helmut: *Linux-Unix-Profitools: awk, sed, lex, yacc und make*. 3., überarb. Aufl. Bonn; Reading, Mass. u.a. : Addison-Wesley-Longman, 1999 (Linux/Unix und seine Werkzeuge). – ISBN 3-8273-1448-8

[Herrmann u. a. 1998] HERRMANN, Uwe ; LENZ, Dierk ; UNBESCHEID, Günter: *Oracle8 für den DBA: Verwalten, optimieren, vernetzen*. Bonn; Reading, Massachusetts u.a. : Addison-Wesley-Longman, 1998. – ISBN 3-8273-1310-0

[Hill u. a. 2001] HILL, Linda L. ; CROSIER, Scott J. ; SMITH, Terence R. ; GOODCHILD, Michael: A Content Standard for Computational Models. In: *D-Lib Magazine* 7 (2001), Nr. 6. – URL http://www.dlib.org/dlib/june01/hill/06hill.html. – ISSN 1082-9873

[Hoch u. a. 1998] HOCH, Ralf ; GABELE, Tobias ; BENZ, Joachim: Towards a standard for documentation of mathematical models in ecology. In: *Ecological Modelling* 113 (1998), S. 3–12

[IEEE 2000a] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (Hrsg.): *IEEE 1516-2000: Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules*. IEEE, 2000. – ISBN 0-7381-2620-9

[IEEE 2000b] Institute of Electrical and Electronics Engineers (Hrsg.): *IEEE 1516.1-2000: Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification*. IEEE, 2000. – ISBN 0-7381-2621-7

[IEEE 2000c] Institute of Electrical and Electronics Engineers (Hrsg.): *IEEE 1516.2-2000: Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specification*. IEEE, 2000. – ISBN 0-7381-2523-3

[IEEE 2003] Institute of Electrical and Electronics Engineers (Hrsg.): *IEEE 1516.3-2003: Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)*. IEEE, 2003. – ISBN 0-7381-3584-4

[IPCC 2001] McCarthy, James J. (Hrsg.) ; Canziani, Osvaldo F. (Hrsg.) ; Leary, Neil A. (Hrsg.) ; Dokken, David J. (Hrsg.) ; White, Kasey S. (Hrsg.): *Climate Change 2001: Impacts, Adaptation and Vulnerability – Contribution of Working Group II to the Third Assessment Report of Intergovernmental Panel on Climate Change*. Cambridge, UK : Cambridge University Press, 2001. – URL http://www.grida.no/climate/ipcc_tar/wg2/. – ISBN 0-521-01500-6

[ISO 1989] Internatinal Organization for Standardization (Hrsg.): *ISO/IEC 2382-7: Information technology – Vocabulary – Part 7: Computer programming*. Geneve, Switzerland : Internatinal Organization for Standardization, 1989

[ISO 1990] Internatinal Organization for Standardization (Hrsg.): *ISO/IEC 2382-20: Information technology – Vocabulary – Part 20: System development*. Geneve, Switzerland : Internatinal Organization for Standardization, 1990

[ISO 1993] Internatinal Organization for Standardization (Hrsg.): *ISO/IEC 2382-1: Information technology – Vocabulary – Part 1: Fundamental terms*. Geneve, Switzerland : Internatinal Organization for Standardization, 1993

[ISO 1998] Internatinal Organization for Standardization (Hrsg.): *ISO/IEC 10746: Reference Model – Open Distributed Processing (RM-ODP), Part 1*. Geneve, Switzerland : International Organization for Standardization, 1998

[ISO 2000] International Organization for Standadization - ISO/TC211 Secretariat (Hrsg.): *ISO/DIS 199115 Geographic Infor-*

mation – Metadata. Version 5. Geneve, Switzerland : Internatinal Organization for Standardization, 2000 Siehe (Kottmann, 2001). – URL http://www.opengis.org/techno/abstract/01-111.pdf. – Project Document Number: 01-111.doc

[ISO 2003] INTERNATINAL ORGANIZATION FOR STANDARDIZATION (Hrsg.): *ISO 15836: Information and Documentation – The Dublin Core metadata element set*. Geneve, Switzerland : Internatinal Organization for Standardization, 2003

[Jaeger u. a. 2002] JAEGER, Carlo C. ; LEIMBACH, Marian ; CARRARO, Carlo ; HASSELMANN, Klaus ; HOURCADE, Jean-Charles ; KEELER, Andrew ; KLEIN, Rupert: Integrated Assessment Modeling: Modules for Cooperation / Fondazione Eni Enrico Mattei. URL http://www.feem.it/Feem/Pub/Publications/WPapers/WP2002-053.htm, July 2002 (NOTA DI LAVORO 53.2002). – Working Paper

[Kainuma u. a. 2003] KAINUMA, Mikiko (Hrsg.) ; MATSUOKA, Yuzuru (Hrsg.) ; MORITA, Tsuneyuki (Hrsg.): *Climate Policy Assessment – Asia-Pacific Integrated Modeling*. Tokyo; Berlin; Heidelberg; New York : Springer, 2003. – ISBN 4-431-70264-4

[Kickert u. a. 1999] KICKERT, Ronald N. ; TONELLA, Giorgio ; SIMONOV, Alexander ; KRUPA, Sagar V.: Predictive modeling of effects under global change. In: *Environmental Pollution* 100 (1999), S. 87–132

[Klein Goldewijk und Battjes 1997] KLEIN GOLDEWIJK, C.G.M. ; BATTJES, J.J.: A Hundred Year (1890 - 1990) Database for Integrated Environmental Assessments (HYDE, version 1.1) / National Institute of Public Health and the Environment (RIVM). February 1997 (Report Nr. 422514002). – Forschungsbericht

[Klein Goldewijk 2001] KLEIN GOLDEWIJK, Kees: Estimating global land use change over the past 300 years: The HYDE Database. In: *Global Biogeochemical Cycles* 15 (2001), June, Nr. 2, S. 417–433

[Kottman 1999] KOTTMAN, Cliff (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 12: OpenGIS Service Architecture*. Version 4. Wayland, Massachusetts, USA : Open GIS Consortium, 1999. – URL http://www.opengis.org/techno/abstract/99-112.pdf. – Project document number: 99-112.doc

[Kottmann 1999a] KOTTMANN, Cliff (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 0: Abstract Specification Overview*. Version 4. Wayland, Massachusetts, USA : Open GIS Consortium, 1999. – URL http:

//www.opengis.org/techno/abstract/99-100r1.pdf. – Project Document Number: 99-100r1.doc

[Kottmann 1999b] Kottmann, Cliff (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 10: Feature Collections.* Version 4. Wayland, Massachusetts, USA : Open GIS Consortium, 1999. – URL http://www.opengis.org/techno/abstract/99-110.pdf. – Project Document Number: 99-110.doc

[Kottmann 1999c] Kottmann, Cliff (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 13: Catalog Services.* Version 4. Wayland, Massachusetts, USA : Open GIS Consortium, 1999. – URL http://www.opengis.org/techno/abstract/99-113.pdf. – Project document number: 99-113.doc

[Kottmann 1999d] Kottmann, Cliff (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 16: Image Coordinate Transformation Services.* Version 4. Wayland, Massachusetts, USA : Open GIS Consortium, 1999. – URL http://www.opengis.org/techno/abstract/99-116r2.pdf. – Project document number: 99-116r2.doc

[Kottmann 1999e] Kottmann, Cliff (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 5: Features.* Version 4. Wayland, Massachusetts, USA : Open GIS Consortium, 1999. – URL http://www.opengis.org/techno/abstract/99-105r2.pdf. – Project Document Number: 99-105r2.doc

[Kottmann 2000] Kottmann, Cliff (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 6: The Coverage Type and its Subtypes.* Version 6. Wayland, Massachusetts, USA : Open GIS Consortium, 2000. – URL http://www.opengis.org/techno/abstract/00-106.pdf. – Project Document Number: 00-106.doc

[Kottmann 2001] Kottmann, Cliff (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 11: OpenGIS^{TM} Metadata (ISO/TC 211 DIS 19115).* Version 5. Wayland, Massachusetts, USA : Open GIS Consortium, 2001. – URL http://www.opengis.org/techno/abstract/01-111.pdf. – Project Document Number: 01-111.doc

[Krück u. a. 2001] Krück, Carsten ; Ploetz, Christiane ; Wiedmann, Thomas ; Zweck, Axel ; BMBF, Bundesministerium für Bildung und F. (Hrsg.): *Forschung zum Globalen Wandel – Wissen für die Zukunft der Erde.* München : Walter Biering GmbH, Mediahaus Grafischer Betrieb, Juni 2001 (BMBF Publik)

[Kuhl u. a. 1999] Kuhl, Frederick ; Weatherly, Richard ; Dahmann, Judith: *Creating computer simulation systems: an indroduction to the high level architecture.* Prentice Hall PTR, 1999

[Kuhn u. a. 2001] KUHN, Werner ; BASEDOW, Sebastian ; BROX, Christoph ; RIEDEMANN, Catharina ; ROSSOL, Holger ; SENKLER, Kristian ; ZENS, Katharina ; LANDES NORDRHEIN-WESTFALEN, Ministerpräsident des (Hrsg.): *GDI Geodaten-Infrastruktur Nordrhein-Westfalen – Referenzmodell 3.0*. Düsseldorf : media NRW, 2001 (media NRW: Band 26)

[Liebl 1995] LIEBL, Franz: *Simulation: problemorientierte Einführung*. 2. überarbeitete Auflage. München; Wien; Oldenbourg : Oldenbourg Verlag, 1995. – ISBN 3-486-23373-4

[Luiten 1999] LUITEN, H.: A legislative view on science and predictive models. In: *Environmental Pollution* 100 (1999), Nr. 100, S. 5–11

[Maxwell 1999] MAXWELL, Thomas: A paris-model approach to modular simulation. In: *Environmental Modelling and Software* 14 (1999), Nr. 6, S. 511–517

[McCarthy u. a. 2001] MCCARTHY, James J. ; CANZIANI, Oswaldo F. ; LEARY, Neil a. ; DOKKEN, David J. ; WHITE, Kasey S.: Climate Change 2001: Impacts, Adaption & Vulnerability – Technical Summary. In: *Climate Change 2001: Impacts, Adaptions, and Vulnerability. Contribution of Working Group II to the Third Assessment Report of the Intergovernmental Panel on Climate Change (IPCC)*. Cambridge, UK : Cambridge University Press, 2001

[Mealling und Denenberg 2002] MEALLING, Michael (Hrsg.) ; DENENBERG, Ray (Hrsg.): *Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names. (URNs): Clarifications and Recommendations*. IETF, August 2002 (RFC 3305). – URL http://www.ietf.org/rfc/rfc3305.txt?number=3305

[Mesarovic u. a. 1996] MESAROVIC, Mihajlo ; MCGINNIS, David L. ; WEST, Dalton A.: Cybernetics of Global Change: Human Dimensions and Managing of Complexity. UNESCO, 1996 (MOST Policy Paper No. 3). – Forschungsbericht

[Moats 1997] MOATS, Ryan: *URN Syntax*. IETF, 1997 (RFC 2141). – URL http://www.ietf.org/rfc/rfc2141.txt?number=2141

[Nakicenovic u. a. 2000] NAKICENOVIC, Nebojsa ; ALCAMO, Joseph ; DAVIS, Gerald ; ET AL.: *Special Report on Emission Scenarios*. Cambridge, UK : Cambridge University Press, 2000. – ISBN 0-521-80493-0

[Nebert 2002] NEBERT, Douglas (Hrsg.): *The OpenGIS® Implementation Specification – Catalog Services*. Version 1.1.1. Wayland, Massachusetts, USA : Open GIS Consortium, 2002. – URL http://www.opengis.org/techno/specs/02-087r3.pdf. – Project document number: OGC 02-087r3

[Nebert 2001] NEBERT, Douglas D. (Hrsg.): *Developing Spatial Data Infrastructures: The SDI Cookbook*. Version 1.1. Reston, VA, USA : The Global Spatial Data Infrastructure Secretariat, 2001. – URL http://www.gsdi.org/pubs/cookbook/cookbook0515.pdf

[New u. a. 1999] NEW, Mark ; HULME, Mike ; JONES, Phil: Representing Twentieth-Century Space-Time Climate Variability. Part I: Development of a 1961-1990 Mean Monthly Terrestrial Climatology. In: *Journal of Climate* 12 (1999), March, S. 829–856

[New u. a. 2000] NEW, Mark ; HULME, Mike ; JONES, Phil: Representing Twentieth-Century Space-Time Climate Variability. Part II: Development of 1961-1990 Monthly Grids of Terrestrial Surface Climate. In: *Journal of Climate* 13 (2000), July, S. 2217–2237

[Oestereich 1998] OESTEREICH, Bernd: *Objektorientierte Softwareentwicklung: Analyse und Desig mit der Unified Modeling Language*. 4., aktualisierte Aufl. München, Wien : Oldenbourg, 1998. – ISBN 3-486-24787-5

[OMG 1999] OBJECT MANAGEMENT GROUP (Hrsg.): *OMG Unified Modeling Language - Specification V.1.3 (June 1999)*. Object Management Group, 1999. – URL www.rational.com/uml/resources/documentation/index.jsp

[Parson 1995] PARSON, Edward A.: Integrated assessment and environmental policy making. In: *Energy Policy* 23 (1995), Nr. 4/5, S. 463–475

[Peirce 1998] PEIRCE, Martin: Computer-Based Models in Integrated Environmental Assessment / European Environmental Agency. European Environmental Agency, 1998 (AEAT-1987). – Technical Report 14. – URL http://reports.eea.eu.int/TEC14/en/tech14.pdf

[Percivall 2002] PERCIVALL, George (Hrsg.): *The OpenGIS^{TM} Abstract Specification – Topic 12: OpenGIS Service Architecture*. Version 4.3. Wayland, Massachusetts, USA : Open GIS Consortium, 2002. – URL http://www.opengis.org/techno/abstract/02-112.pdf. – Project document number: 02-112.doc

[Poetzsch-Heffter 2001] POETZSCH-HEFFTER, A.: *Software-Architektur*. Bd. 1: *Architektur von Softwaresystemen*. Hagen : FernUniversität - Gesamthochschule Hagen, 2001

[Ramachandran u. a. 2003] RAMACHANDRAN, Rahul ; CONOVER, Helen ; GRAVES, Sara ; CHRISTOPHER, Sundar: EARTH SCIENCE MARKUP LANGUAGE: A Solution to Earth Science Data Format Heterogeneity Problem. In: *American Meteorological Society's (AMS) 19th International Conference*

on Interactive Information Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology, Long Beach, CA, Feb. 9 - 13, 2003, URL http://ams.confex.com/ams/annual2003/techprogram/paper_54086.htm, Februry 2003

[Rehesaar 1996] REHESAAR, Hugo: International Standards: Practical or just theoretical? In: *StandardView* 4 (1996), September, Nr. 3, S. 123–126

[Rizzoli und Davis 1999] RIZZOLI, Andrea E. ; DAVIS, J. R.: Integration and re-use of Environmental Models. In: *Environmental Modelling & Software* 14 (1999), Nr. 6, S. 493–494 (Editorial)

[Roehrl und Schmiedl 2002] ROEHRL, Armin ; SCHMIEDL, Stefan: Die wichtigsten Open-Source-Lizenzen. In: *c't Magazin für Computer-Technik* 1 (2002), S. 170ff

[Rotmans u. a. 1994] ROTMANS, J. ; ASSELT, M.B.A. van ; BRUIN, A.J. de ; ELZEN, M.G.J. den ; GREEF, J. de ; HILDERINK, H. ; HOEKSTRA, A.Y. ; JANSSEN, M.A. ; KOSTER, H.W. ; MARTENS, W.J.M. ; NIESSEN, L.W. ; VRIES, H.J.M. de: *Global Change and Sustainable Development: A Modeling Perspective for the Next Decade*. Bilthoven, The Netherlands : National Institute of Public Health and Environmental Protection (RIVM), Juni 1994. – URL http://sedac.ciesin.org/mva/JR1994A/JR1994A.html

[Rotmans 1998] ROTMANS, Jan: Methods for IA: The challanges and opportunities ahead. In: *Environmental Modeling and Assessment* (1998), Nr. 3, S. 155–179

[Rotmans und van Asselt 2001] ROTMANS, Jan ; ASSELT, Marjolein B. A. van: Uncertainty management in integrated assessment modelling: towards a pluralistic approach. In: *Environmental Monitoring and Assessment* (2001), Nr. 69, S. 101–130

[Rotmans und Dowlatabadi 1998] ROTMANS, Jan ; DOWLATABADI, Hadi: Integrated assessment modeling. In: RAYNER, Steve (Hrsg.) ; MALONE, Elizabetz L. (Hrsg.): *Human choice and climate change* Bd. 3 – The tools for policy analysis. Columbus, Ohio, USA : Battelle Press, 1998, Kap. 5, S. 291–377

[Schneider 1997] SCHNEIDER, Stephen H.: Integrated assessment modeling of global climate change: Transparent rational tool for policy making or opaque screen hiding value-laden assumptions? In: *Environmental Modeling and Assessment* (1997), Nr. 2, S. 229–249

[Schulze u. a. 1999] Schulze, Thomas ; Strassburger, Steffen ; Klein, Ulrich: Migration of HLA into Civil Domains: Solutions and Prototypes for Transportation Applications. In: *Simulation* 73 (1999), Nr. 5, S. 296–303. – URL http://isgsim1.cs.uni-magdeburg.de/hla/paper/S7305-05.pdf. – ISSN 0037-5497/99

[Schürmann 1995] Schürmann, Gerd: The evolution from open systems interconnection (OSI) to open distributed processing (ODP). In: *Computer Standards & Interfaces* 17 (1995), January, Nr. 1, S. 107–113

[Schwinn 1992] Schwinn, Hans: *Relationale Datenbanksysteme.* München; Wien : Hanser, 1992 (Hanser Studienbücher der Informatik). – ISBN 3-446-15782-4

[Senkler 2001] Senkler, Kristian: Anforderungen an eine offene Infrastruktur für E-Business und Geoinformations-Services. In: Strobl, Josef (Hrsg.) ; Blaschke, Thomas (Hrsg.) ; Griesebner, Gerald (Hrsg.): *Angewandte Geographische Informationsverarbeitung XIII. Beiträge zum AGIT-Symposium Satzburg 2001.* Heidelberg : Wichmann Verlag, 2001, S. 455–460

[Shaw und Garlan 1996] Shaw, Mary ; Garlan, David: *Software-Architecture. Perspectives on an Emerging Discipline.* Upper Saddle River, New Jersey : Prentice Hall - Alan Apt, 1996

[Shlyakhter u. a. 1995] Shlyakhter, Alexander ; Valverde A. Jr, L. J. ; Wilson, Richard: Integrated risk analysis of global climate change. In: *Chemosphere* 30 (1995), Nr. 8, S. 1585–1618

[Steinhausen 1994] Steinhausen, Detlef: *Simulationstechniken.* München, Wien : Oldenbourg, 1994

[Swoboda u. a. 2000] Swoboda, Walter ; Kruse, Fred ; Legat, Rudolf ; Nikolai, Ralf ; Behrens, Sven: Harmonisierter Zugang zu Umweltinformationen für Öffentlichkeit, Politik und Planung: Der Umweltdatenkatalog UDK im Einsatz. In: Cremers, Armin B. (Hrsg.) ; Greve, Klaus (Hrsg.): *Umweltinformatik '00, Umweltinformation für Planung, Politik und Öffentlichkeit. 14. Internationales Symposium „Informatik für den Umweltschutz" der Gesellschaft für Informatik (GI), Bonn 2000* Bd. 2. Marburg : Metropolis Verlag, 2000, S. 595–607. – URL http://www.umweltdatenkatalog.de/publikat/pdfs/harmonisierterzugang.pdf

[Swoboda u. a. 1998] Swoboda, Walter ; Kruse, Fred ; Nyhuis, Detlev ; Rousselle, Holger: Die Neukonzeption des Umweltdatenkataloges. In: Haasis, Hans-Dietrich (Hrsg.) ; Ranze, K. C. (Hrsg.): *Umweltinformatik '98, Vernetzte Strukturen in Informatik, Umwelt und Wirtschaft. 12. „Informatik für den Umweltschutz" der Gesellschaft für Informatik (GI), Bremen 1998*

Bd. 2. Marburg : Metropolis Verlag, 1998, S. 610–620. – URL http://www.umweltdatenkatalog.de/publikat/pdfs/NeukonzeptiondesUDK.pdf

[Thomas und Nejmeh 1992] Thomas, Ian ; Nejmeh, Brian A.: Definitions of Tool Integration for Environments. In: *IEEE Software* 9 (1992), March/April, Nr. 2, S. 29–35

[Tol und Vellinga 1998] Tol, Richard S. J. ; Vellinga, Pier: The European Forum on Integrated Environmental Assessment. In: *Environmental Modeling and Assessment* (1998), Nr. 3, S. 181–191

[Toth 1995] Toth, Ferenc L.: Practice and progress in integrated assessment of climate change – A workshop overview. In: *Energy Policy* 23 (1995), Nr. 4/5, S. 253–267

[UN 1992] United Nations (UN) ; Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit (BMU) (Hrsg.): *Konferenz der Vereinten Nationen für Umwelt und Entwicklung im Juni 1992 in Rio de Janeiro. Dokumente: Agenda 21.* Bonn : Köllen Druck+Verlag, 1992 (Umweltpolitik). – URL http://www.bmu.de/download/dateien/agenda21.pdf

[van der Sluijs 1996] van der Sluijs, Jeroen: Integrated assessment models and the management of uncertainties / IIASA (International Institute for Appliend Systems Analysis). Laxenburg, Austria : IIASA (International Institute for Appliend Systems Analysis), October 1996. – Working Paper WP-96-119

[van Wijk 1994] van Wijk, Jack: *M-Software Architecture.* Documentation available at M web page (www.m.rivm.nl). Juni 1994

[Vienneau 2001] Vienneau, Aleta: *Using ArcCatalog^{TM}.* USA : ESRI®, Environmental System Research Institute, 2001. – ISBN ISBN: 1879102951

[Villa 2001] Villa, Ferdinando: Integrating modelling architecture: a declarative framework for multi-paradigm, multi-scale ecological modelling. In: *Ecological Modelling* 137 (2001), Nr. 1, S. 23–42

[Villa und Costanza 2000] Villa, Ferdinando ; Costanza, Robert: Design of multi-paradigm integrating modelling tools for ecological reserach. In: *Environmental Modelling and Software* (2000), Nr. 15, S. 169–177

[Voges 2001] Voges, Uwe: terraSeek – OpenGIS basierter Katalogdienst für Geodaten und Geoservices. In: Hilty, Lorenz M. (Hrsg.) ; Gilgen, Paul W. (Hrsg.): *Sustainability in the Information Society. 15th international Symposium Informatics for Environmental Protection, Zurich 2001* Bd. 1: Impacts and Applications. Marburg : Metropolis Verlag, 2001, S. 484–492

[WBGU 1993] Wissenschaftlicher Beirat der Bundesregierung Globale Umweltveränderungen (Hrsg.): *Welt im Wandel: Grundstruktur globaler Mensch-Umwelt-Beziehungen; Jahresgutachten 1993*. Bonn : Economica Verlag, 1993. – URL http://www.wbgu.de/wbgu_jg1993.pdf

[Welsh u. a. 2000] Welsh, Matt ; Dalheimer, Matthias K. ; Kaufman, Lar: *LINUX – Wegweiser zur Installation & Konfiguration*. Köln : O'Reilly, 2000. – ISBN 3-89721-133-5

[Weyant u. a. 1996] Weyant, J. ; Davidson, O. ; Dowlatabadi, H. ; Edmonds, J. ; Grubb, M. ; Parson, E.A. ; Richels, R. ; Rotmans, J. ; Shukla, P.R. ; Tol, R.S.J. ; Cline, W. ; Fankhauser, S.: Integrated Assessment of Climate Change: An Overview and Comparison of Approaches and Results. In: IPCC, (Intercovernmental Panel on Climate Change) (Hrsg.): *Climate Change 1995*. Cambridge, UK : Cambridge University Press, 1996, S. 369–396

Anhang A

Glossar

Architektur Siehe *Software-Architektur*.

Assessment-Modell Siehe *Integriertes Assessment-Modell*.

Daten Eine wieder interpretierbare Darstellung von Information in formalisierter Weise, geeignet zur Übermittlung, Deutung oder Verarbeitung. (ISO, 1993; Quelle: DIN, 1995)

Datensatz-Serie Sammlung von Datensätzen mit gleicher Produktspezifikation. (**?**)

Datenbank Die Gesamtheit der Daten eines Anwendungsbereichs. (Balzert, 1996)

Datenbank-Managementsystem (DBMS) System zur zentralen Verwaltung und Kontrolle der Datenbestände von Datenbanken. (Balzert, 1996)

Datenbanksystem System für die dauerhafte, zuverlässige und unabhängige Verwaltung sowie die komfortable, flexible und geschützte Verwendung großer, integrierter und mehrfachbenutzbarer Datenbanken. (Balzert, 1996)

Datenhaltungssystem Siehe Datenbanksystem.

Dienst Abgrenzbarer (distinct) Teil einer Funktionalität, die von einer Entität über eine Schnittstelle bereitgestellt wird. (ISO/IEC TR 14252)

Framework Ein durch den Software-Entwickler anpassbares oder erweiterbares System kooperierender Klassen bei dem in der Regel abstrakte oder leere Operationen in Unterklassen definiert bzw. implementiert werden. (Balzert, 1996)

Globaler Wandel Die Veränderungen in Natur und Gesellschaft, die die Menschheit als Ganzes und auf längere Sicht hin betreffen. (Krück u. a., 2001)

Hilfs-Unterprogramm, Hilfsprogramm Ein Unterprogramm oder ein Programm, das allgemeine, häufig benötigte Dienste für Benutzer und Bedienungspersonal liefert. (ISO, 1989; Quelle: DIN, 1995)

IAM Siehe *Integriertes Assessment-Modell.*

Integriertes Assessment Prozess, in dem Wissen unterschiedlicher Fachdisziplinen über das 'System Erde' in einem konsistenten Rahmen kombiniert und interpretiert wird und der das Ziel verfolgt, den Zustand und mögliche langfristige Änderungen des Systems einzuschätzen und zu bewerten sowie die Ergebnisse politischen Entscheidungsträgern zu vermitteln. (s. Unterabschnitt 2.1.2, Seite 11)

Integrated Assessment Model (IAM) IAMs use a computer program to link an array of component models based on mathematical representations of information from the various contributing disciplines. (Weyant u. a., 1996) Siehe auch den weiter gefassten Begriff 'SISA'.

Integriertes Modell Siehe *Integriertes Assessment-Modell.*

Interoperabilität Eignung, mit vorgegebenen Systemen zusammenzuwirken. (DIN, 1994)

Katalog Eine Sammlung von Katalogeinträgen, die so organisiert ist, dass sie einem Nutzer bei der Suche nach und beim Zugriff auf Ressourcen unterstützt. (in Anlehnung an Kottmann, 1999c13)

Komponente Ein abgeschlossener, binärer Software-Baustein, der eine anwendungsorientierte, semantisch zusammengehörende Funktionalität besitzt, die nach außen über Schnittstellen zur Verfügung gestellt wird. (Balzert, 2000) Baustein für die Struktur eines Systems. (Buschmann u. a., 1998)

Metadaten Daten über den Inhalt, die Qualität, den Zustand und andere Charakteristiken von Daten. (FGDC, 1998)

Programm Eine syntaktische Einheit, die die Regeln einer bestimmten Programmiersprache befolgt und die aus Deklarationen und Anweisungen oder Instruktionen zusammengesetzt ist, notwendig zur Lösung einer gewissen Funktion, Aufgabe oder eines Problems. (ISO, 1993; Quelle: DIN, 1995)

Ressource Natürlich vorhandener Bestand von etwas, was für einen bestimmten Zweck benötigt wird. (Duden, 1996) ("source of supply or support; a source of information or expertise" Merriam-Webster´s Collegiate Dictionary, http://www.britannica.com.)

Schnittstelle Benannter Satz von Operationen, der das Verhalten einer Entität charakterisiert. (ISO/DIS 19119)

Service Siehe *Dienst*

Simulation/Simulationslauf Durchführung einer Modellberechnung unter definierten Bedingungen. (vgl. Simulationslauf-Spezifikation)

Simulationslauf-Spezifikation Definierte Modell-Konfiguration. Beinhaltet die Spezifikation aller Simulationsmodell-Daten (inkl. Systemparameter, Modellumwelt-Größen, Optionen). (s. Unterabschnitt 4.1.2, Seite 72)

Simulationsmodell „A set of computational codes, executable in some software/hardware environment, that transform a set of input data into a set of output data, with the input, output, and transformation typically having some interpretation in terms of real-world phenomena." Definition des Begriffs 'computational model' nach Hill u. a. (2001).

SISA Siehe *System zum integrierten simulationsbasierten Assessment*

Software Programme, Verfahren, Regeln und möglicherweise zugehörige Dokumentationen und Daten zum Betrieb eines Computersystems. (IEEE 610.12)

Software-Architektur Eine Software-Architektur ist die grundsätzliche Strukturierung eines Software-Systems. Sie beschreibt eine Menge definierter Komponenten, die über Schnittstellen miteinander kommunizieren, spezifiziert deren jeweiligen Zuständigkeitsbereich und beschreibt die Beziehungen zwischen den Komponenten. (s. Unterabschnitt 2.3.2, Seite 18)

Software-Erzeugnis Eine vollständige und dokumentierte Menge von Programmen zur Lieferung an mehrere Benutzer für eine Anwendungs- oder Funktionsgattung. (ISO, 1990; Quelle: DIN, 1995)

Software-System Eine Menge von Software-Komponenten, die gemeinsam eine oder mehrere Aufgaben erfüllen. (Buschmann u. a., 1998)

Subsystem Eine Menge von miteinander arbeitenden *Komponenten*, die gemeinsam eine Aufgabe erfüllen und die als eigenständige Einheit innerhalb einer *Software-Architektur* angesehen werden wird. (Buschmann u. a., 1998)

System zum integrierten simulationsbasierten Assessment (SISA) Ein System zum integrierten simulationsbasierten Assessment (SISA) ist ein Software-System, das von unterschiedlichen Fachdisziplinen stammende Daten und Simulationsmodelle zum 'System Erde' in einem konsistenten Rahmen kombiniert und neue Daten über den Zustand und mögliche langfristige Änderungen des 'Systems Erde' – vornehmlich zur Unterstützung politischer Entscheidungsträger – berechnet und bereitstellt. (s. Unterabschnitt 3.1, Seite 25)

Szenario In sich konsistente und plausible Annahmen über die zukünftige Entwicklung systembeeinflussender exogener Größen. (Bossel, 1994)

Anhang B

Datenmodelle und Schnittstellen

B.1 Datenmodell zu Personen und Organisationen

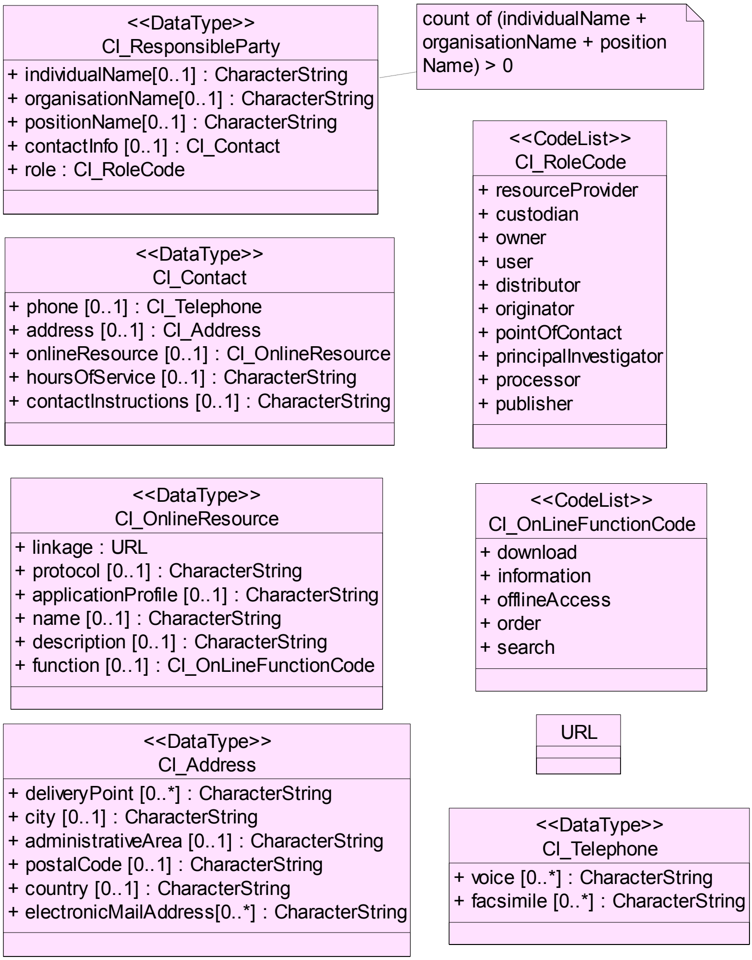


Abbildung B.1: Zur Verwaltung von Personen und Organisationen (Institutionen) verwendete Klassen nach ISO 19115 (Responsible party information). Nähere Informationen finden sich in Unterabschnitt 5.2.3 (Seite 119). Quelle: Kottmann (2001).

B.2 Zusammenfassung des SISA-Datenmodells

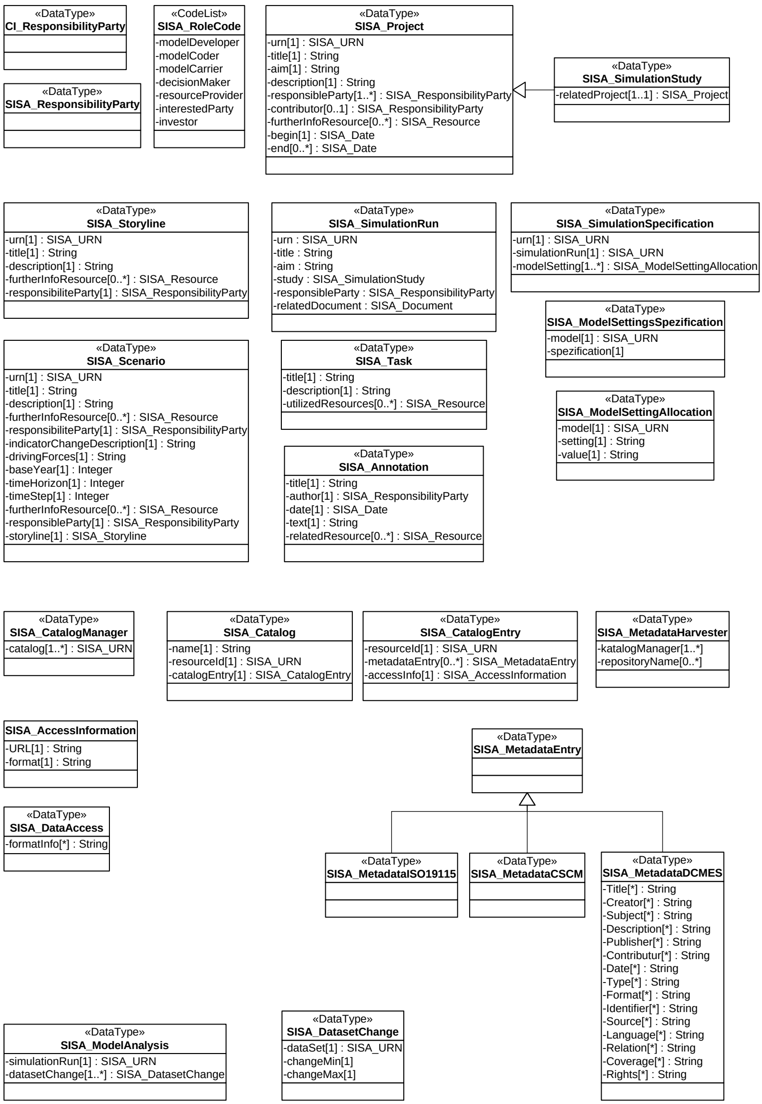


Abbildung B.2: Übersicht des SISA-Datenmodells. Erklärungen finden sich in Abschnitt 5.2 (Seite 96).

B.3 Zusammenfassung der SISA-Schnittstellen

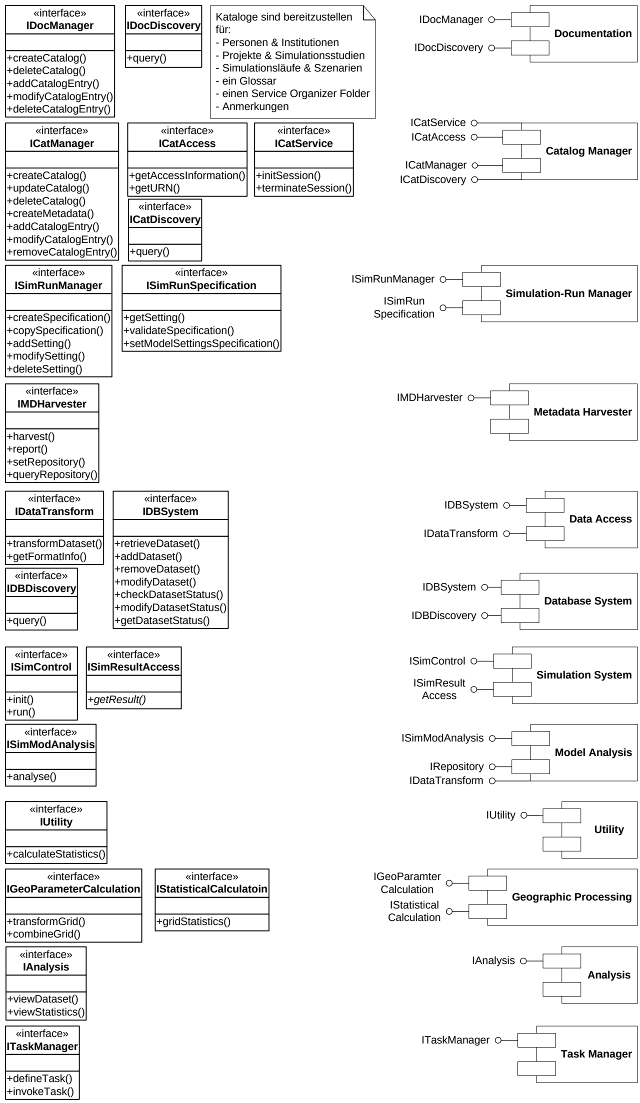


Abbildung B.3: Übersicht der SISA-Koponenten und Schnittstellen. Erklärungen finden sich in Abschnitt 5.2 (Seite 96).

Anhang C

Standards

C.1 Übersichten

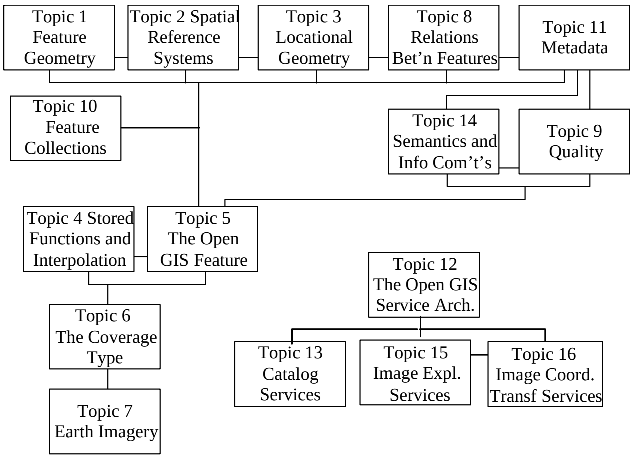


Abbildung C.1: Überblick über die Abhängigkeiten der OpenGIS Abstract Specifications untereinander. Quelle: Kottmann (1999a).

Dienste-Taxonomie der ISO/DIS 19119, Teil I

Taxonomie-Klasse	Beispiel
Geographic Human Interaction Services	catalogue viewer geographic viewer geographic viewer – animation geographic viewer – mosaicing geographic viewer – perspective geographic viewer – imagery geographic spreadsheet viewer service editor chain definition editor workflow enactment manager geographic feature editor geographic symbol editor feature generalization editor geographic data-structure viewer
geographic model/information management services	feature access service map access service coverage access service coverage access service – sensor sensor description service product access service feature type servcie catalogue servcie registry servcie gazetteer service order handling service standing order service
geographic workflow/task management services	chain definition service workslow enactment service subscription service
geographic communication services	encoding service transfer service geographic compression service geographic format conversion service messaging service remote file and executable management
geographic system management services	no services have been identified.

Tabelle C.1: Beipieldienste der ISO/DIS 19119 Taxonomie, Teil I.

.

Dienste-Taxonomie der ISO/DIS 19119, Teil II

Taxonomie-Klasse	Beispiel
geographic processing services – spatial	coordinate conversion service coordinate transformation service coveratge/vector conversion service image coordinate conversion service rectification service sensor geometry model adjustment service image geometry model conversion service subsetting service sampling service tiling change service dimension measurement service feature manipulation service feature matching service feature generalization service route determination service positioning service proximity analysis servcie
geographic processing services – thematic	geoparameter calculation service thematic classification service feature generalization service subsetting service spatial counting service change detection service geographic information extraction service image processing service reduced resolution generation service image manipulation service image understnading service image synthesis service multi-band image manipulation service object detection service geocoding service
geographic processing services – temporal	temp. reference system transformation s. subsetting service sampling service temporal proximity analysis service
geographic processing services – metadata	statistical calculation service geographic annotation service

Tabelle C.2: Beipieldienste der ISO/DIS 19119 Taxonomie, Teil II.

.

Standards aus der ISO-19100-Reihe

Bezeichnung	Thema
ISO 19101	Reference model
ISO 19102	Overview
ISO 19103	Conceptual schema language
ISO 19104	Terminology
ISO 19105	Conformance and testing (published)
ISO 19106	Profiles
ISO 19107	Spatial schema
ISO 19108	Temporal schema
ISO 19109	Rules for application schema
ISO 19110	Feature cataloguing methodology
ISO 19111	Spatial referencing by coordinates
ISO 19112	Spatial referencing by geographic identifiers
ISO 19113	Quality principles
ISO 19114	Quality evaluation procedures
ISO 19115	Metadata
ISO 19116	Positioning services
ISO 19117	Portrayal
ISO 19118	Encoding
ISO 19119	Services
ISO/TR 19120	Functional standards + new revision started
ISO/TR 19121	Imagery and gridded data (published
ISO/TR 19122	Qualifications and certification of personnel
ISO 19123	Schema for coverage geometry and functions
ISO 19124	Imagery and gridded data components
ISO 19125-1	Simple feature access - Part 1: Common architecture
ISO 19125-2	Simple feature access - Part 2: SQL option
ISO 19125-3	Simple feature access - Part 3: COM/OLE option
ISO 19126	Profile - FACC Data Dictionary
ISO 19127	Geodetic codes and parameters
ISO 19128	Web map server interface
ISO 19129	Imagery, gridded and coverage data framework
ISO 19130	Sensor and data models for imagery and gridded data
ISO 19131	Data product specifications
ISO 19132	Location based services possible standards
ISO 19133	Location based services tracking and navigation
ISO 19134	Multimodal location based services for routing and navigation
ISO 19135	Procedures for registration of geographical information items
ISO 19136	Geography Markup Language
ISO 19137	Generally used profiles of the spatial schema and of similar important other schemas
ISO 19138	Data quality measures
ISO 19139	Metadata - Implementation specification
ISO 19140	Technical amendment to the ISO 191** Geographic information series of standards for harmonization and enhancements

Tabelle C.3: Standards im Arbeitsprogramm des ISO/TC 211 (Stand: Januar 2003, Quelle: `http://www.isotc211.org`). Aufgeführt sind alle Standards, unabhängig vom aktuellen Status (der vom ersten *Working Draft* bis zum *International Standard* reicht). Siehe auch Abschnitt 32 (Seite 42).

Abstract Specifications des OpenGIS-Konsortiums

Topic-Nr.	Topic-Name	Version
0	Overview	4
1	Feature Geometry[a]	5
2	Spatial Reference Systems[b]	1
3	Locational Geometry	4
4	Stored Functions and Interpolation	4
5	The OpenGIS Feature	4
6	The Coverage Type	6
7	Earth Imagery[c]	4
8	Relations Between Features	4
9	Accuracy [d]	4
10	Feature Collections	4
11	Metadata[e]	5
12	The OpenGIS Service Architecture[f]	4.3
13	Catalog Services	4
14	Semantics and Information Communities	4
15	Image Exploitation Services	6
16	Image Coordinate Transformation Services	4

[a] Harmonisierung mit ISO in 2001: ISO 19119 wurde übernommen. Aktuelle Version enthält ISO/DIS 19107 Spatial Schema

[b] Basis ist ISO 19111, es wurden allerdings Erweiterungen und Änderungen vorgenommen. Die Unterschiede zu ISO/DIS 19111 werden im Anhang zu Topic 2 zusammengefasst

[c] Entwicklung in enger Kooperation mit ISO/TC 211

[d] nächste Version dieses Dokuments wird wahrscheinlich auf Arbeiten zum Qualitäts-Thema der des ISO/TC 211 aufbauen

[e] Harmonisierung mit ISO in 2001: ISO/DIS 19115 wurde als OGC Topic 11 übernommen

[f] Harmonisierung mit ISO in 2001: ISO 19119 wurde übernommen. Aktuelle Version enthält den Inhalt von ISO/DIS 19119

Tabelle C.4: Auflistung verfügbarer OpenGIS Abstract Specifications mit Anmerkungen zu Haronisierungsbestrebungen mit der ISO (Stand: Januar 2003, Quelle: http://www.opengis.org).

Implementation Specifications des OpenGIS-Konsortiums

Titel	Version
OpenGIS Simple Features Specification for OLE/COM	1.1
OpenGIS Simple Features Specification for CORBA	1.0
OpenGIS Simple Features Specification for SQL	1.1
OpenGIS Catalog Services Implementation Specification	1.1.1
OpenGIS Grid Coverages Implementation Specification	1.0
OpenGIS Coordinate Transformation Services Implementation Specification	1.0
OpenGIS Web Map Server Interfaces Implementation Specification	1.1.1
OpenGIS Geography Markup Language (GML) Implementation Specification	2.1.2
OpenGIS Web Feature Service Implementation Specification	1.0
OpenGIS Filter Encoding Implementation Specification	1.0
OpenGIS Styled Layer Descriptor Implementation Specification	1.0

Tabelle C.5: Auflistung verfügbarer OpenGIS Implementation Specifications (Stand: Januar 2003, Quelle: http://www.opengis.org).

C.2 HLA-Regeln

Die HLA Rules definieren insgesamt zehn Regeln (fünf für Federations und fünf für Federates):

Regeln für Federations:

1. Federations sollen ein gemäß der HLA Object Model Template (OMT) dokumentiertes HLA federation object model (FOM) besitzen.
2. Innerhalb einer federation sollen alle simulationsbezogenen Repräsentationen von Objekt-Instanzen in den federates sein und nicht in der runtime infrastructure (RTI).
3. Während der Ausführung einer federation soll der gesamte Austausch von FOM-Daten zwischen den federates über die RTI erfolgen.
4. Während der Ausführung einer federation sollen die federates gemäß der HLA interface specification mit der RTI interagieren.
5. Während der Ausführung einer federation soll ein Instanz-Attribut zu jeder Zeit Eigentum[1] von maximal einem federate sein.

Regeln für Federates:

6. Federates sollen ein gemäß HLA OMT dokumentiertes HLA simulation object model (SOM) besitzen.
7. Federates sollen in der Lage sein jedes Attribut gemäß der Beschreibung in den SOMs zu aktualisieren und/oder widerzuspiegeln und Interaktionen zu senden und/oder zu empfangen.
8. Federates sollen in der Lage sein den Besitz eines Attributes gemäß der Beschreibung in den SOMs dynamisch, d. h. zur Laufzeit der Federation, zu übertragen oder anzunehmen.
9. Federates sollen im Einklang mit den Beschreibungen in den SOMs in der Lage sein die Bedingungen (z. B. Grenzwerte) zu verändern unter denen sie Aktualisierungen (Updates) der Attribute bereitstellen.
10. Federates sollen in der Lage sein die lokale Zeit in einer Art und Weise zu verwalten, die ihnen einen koordinierten Datenaustausch mit anderen Mitgliedern der federation erlaubt.

[1] Eigentum/Besitz (ownership) ist hier definiert als die Berechtigung, die Attribut-Werte einer Instanz zu aktualisieren.

Anhang D

Programm-Quelltexte

D.1 PHP-Beispiel

Datenbankverbindung

Der folgenden PHP-Quelltext (dbconnect.php) wird beim Aufruf jeder PHP-Seite mit Datenbank-Funktionalität aufgerufen.

```
<!-- dbconnect.php -->
<?php
    if (!($mylink = mysql_connect("localhost","rglassUser",
          "rglassPasswd")))                                   (1)
    {
        print "<h3>Keine Verbindung zur Datenbank!</h3>\n";
        exit;                                                 (2)
    }
    mysql_select_db("rglassDB");                              (3)
?>
```

Zunächst wird versucht eine Verbindung zum Datenbank-Server herzustellen – s. Punkt (1). Für diesen Aufruf wird sowohl die Internet-Adresse des Servers (hier 'localhost', also der lokale Rechner) als auch der Benutzername ('rglassUser') und dessen Passwort ('rglassPasswd') benötigt. Bei erfolgreicher Verbindung wird die gewünschte Datenbank ausgewählt ('rglassDB') (3), ansonsten wird eine Fehlermeldung ausgegeben (2).

Datenbankabfrage und Datenvisualisierung

Der folgende PHP-Quelltext (viewResponsibleParty.php) dient der Auflistung aller Personen/Organisationen, die in der Datenbank gespeichert sind.

```
<!-- viewResponsibleParty.php -->
<?phpinclude("dbconnect.php");?>                                  (1)
<title>View Responsible Parties</title>
<body bgcolor="#FFFF00">
<h2>List of Responsible Parties</h2>                              (2)

<?php
//----- get data from MySQL database                              (3)
$result = mysql_query("select * from responsibleparty") or die
("Error");

//----- display each dataset in a separate table                  (4)
if ($result) {
  while ($row = mysql_fetch_array($result))
  {

  echo "<table width=90% border=1>\n";
    //----- first database field to display for each
    //----- entry (individual name)                               (5)
    echo "<tr>\n";
      echo "<td width=19%>";
    print "<b>Individual Name: </b>";
      echo "</td>";
      echo "<td  width=22%>";
    print $row["individualName"];
      echo "</td>";
    echo "</tr>\n";

    echo "<tr align=left>\n";
      echo "<td width=19%>";
    print "<b>Organisation Name: </b>";
      echo "</td>";
      echo "<td  width=22%>";
    print $row["organisationName"];
      echo "</td>";
    echo "</tr>\n";

    //...

    //----- last database field to display for each entry         (6)
    echo "<tr align=left>\n";
      echo "<td width=19%>";
    print "<b>E-Mail: </b>";
      echo "</td>";
      echo "<td  width=22%>";
        print "<a href=\"mailto:";
```

```
    print $row["email"];
        print "\">";
        print $row["email"];
      echo "</td>";
    echo "</tr>\n";

  echo "</table>\n";

  print "<br>\n";
  print "<br>\n";
  }
  mysql_free_result($result);
} ?>

<h2><a href="insertResponsibleParty.php">Insert another
party</a></h2>                                                  (7)
```

Zunächst wird über die oben beschriebene Funktionsfolge (dbconnect.php) die Verbindung zur Datenbank hergestellt (1). Nach der Ausgabe einer Überschrift (2) erfolgt dann die Abfrage der Datenbank über einen SQL-Befehl, der für die Selektion aller Einträge der Tabelle ‘responsibleparty’ sorgt (3). Jeder der gefundenen Datensätze wird anschließend über eine Schleife in einer eigenen Tabelle auf dem Bildschirm angezeigt (4) – vom ersten (5) bis zum letzten gewünschten Eintrag (6). Als Abschluss folgt noch ein Verweis auf die Seite zum Eintragen neuer Personen/Organisationen (7). Über die einzelnen PHP-Anweisungen wird zur Laufzeit der HTML-Quelltext erzeugt, der zur Anzeige im Web-Browser benötigt wird.

D.2 Metadaten-Datei

Nachfolgend findet sich der Quelltext einer Metadaten-Datei, die automatisch durch die Web-Seite zur Metadaten-Erfassung erzeugt wurde (vgl. Abbildung 6.10, Seite 181).

```
<?xml version="1.0"?> <?xml-stylesheet type="text/xsl"

href="http://www.usf.uni-kassel.de/grid/meta/metadata.xsl" ?>       (1)
<rdf:rdf xmlns:rdf="http://www.w3.org/rdf/rdf/"                     (2)
         xmlns:dc="http://purl.oclc.org/dc/">                       (3)
  <rdf:Description about=
   "urn:x-wzusf:doc-rep.rglassSummaryReport-V1,0.doc.hd">           (4)
    <!-- Dublin Core Elements generated via metaDataInput.htm -->
    <dc:title> Will Climate Change Affect Food and Water Security
       in Russia? </dc:title>
    <dc:creator> Joseph Alcamo </dc:creator>                        (5)
```

```
    <dc:creator> Genady Golubev </dc:creator>
    <dc:creator> Nikolai Dronin </dc:creator>
    <dc:creator> Andrei Kirilenko </dc:creator>
    <dc:creator> Marcel Endejan </dc:creator>
    <dc:subject> R-GLASS </dc:subject>
    <dc:subject> GLASS </dc:subject>
    <dc:subject> Russia </dc:subject>
    <dc:description> Will Climate Change Affect Food and Water
      Security in Russia? </dc:description>
    <dc:description> Summary Report of the International Project
      on Global Environmental Change and its Thread to Food and
      Water Security in Russia </dc:description>
    <dc:publisher> CESR </dc:publisher>
    <dc:contributor> Guenther Fischer </dc:contributor>
    <dc:date> 2003-03-24 </dc:date>
    <dc:type> Document </dc:type>
    <dc:type> Report </dc:type>
    <dc:format> MS-Word </dc:format>
    <dc:identifier>url:file://usf1/home/_GRID/Marcel/Moscow/
      final_report/rglass_final_2003-03-24.doc
    </dc:identifier>
    <dc:identifier>urn:x-wzusf:doc-rep.rglassSummaryReport-V1,0.doc.hd
    </dc:identifier>
    <dc:source> </dc:source>
    <dc:language>en (also in Russian language available) </dc:language>
    <dc:relation> </dc:relation>
    <dc:coverage> temporal: 1961-1990, 2020s, 2070s </dc:coverage>
    <dc:coverage> spatial: 89 Russian Regions </dc:coverage>
    <dc:rights> </dc:rights>
  </rdf:Description>
</rdf:rdf>
```

Die Datei beschreibt die Ressource 'urn:x-wzusf:doc-rep.rglassSummaryReport-V1,0.doc.hd' – siehe Punkt (4). Hierzu werden die Elemente des *Dublin Core Metadata Element Set* verwendet (Namensraum-Kennzeichnung 'dc') (3), die wiederum eingebettet sind in die Struktur des *Resource Description Framework* (Kennzeichnung 'rdf') (2).

Wie an Punkt (5) in der Datei zu sehen ist, können einige der Elemente mehrfach vorkommen (z. B. <dc:creator> zur Beschreibung der Autoren der Studie), während einige keine Einträge enthalten (z. B. <dc:source>).

D.3 Metadaten-Sammler

Skript für einfachen Harvester unter der Bash-Shell

Das folgende Skript ist ein Beispiel für einen einfachen Metadaten-Sammler unter der *Bourne-Again-Shell (bash).*

```
path=/grid/meta                                                          (1)
# copy old repository mv $path/repository.xml
$path/repository.old

# search for *.dc.xml files and copy content to repository.xml
find $path -name '*.dc.xml' -exec cat {} >  $path/repository.xml \;  (2)
# if necessary add further locations to search path here:
# find nextPath -name '*xml' -exec cat {} >> $path/repository.xml \;

# remove tags not needed
sed -n '/?xml/!p' $path/repository.xml > $path/tmp1.xml                 (3)
sed -n '/xmlns/!p'      $path/tmp1.xml > $path/tmp2.xml
sed -n '/rdf:rdf/!p'    $path/tmp2.xml > $path/tmp3.xml

# add needed tags for RDF and XSL
cat $path/rdfHeader.txt >  $path/repository.xml                          (4)
cat $path/tmp3.xml >>      $path/repository.xml
cat $path/rdfFooter.txt >> $path/repository.xml

# clean files
/bin/rm $path/tmp1.xml $path/tmp2.xml $path/tmp3.xml

# copy repository to www
ftp -i -n < $path/copyRepositoryToWWW.ftp                                (5)
```

Das Skript durchsucht die unter Punkt (1) spezifizierten Verzeichnisse über den find-Befehl (2) und kopiert dabei den kompletten Inhalt der Dateien in eine separate Datei. Anschließend werden redundante XML-Textbegrenzungen (Tags) entfernt (3) und für die Gesamtdatei benötigte Tags hinzugefügt (4).

Die auf diese Weise erzeugte Datei wird dann über das *File Transfer Protocol (ftp)* an einen zentralen Ort kopiert (5) – in diesem Fall ist der zentrale Ort ein Verzeichnis auf dem Web-Server.

Nachteil dieser ersten Version des Metadaten-Sammlers ist die Festlegung auf das starre Format, in dem die Dateien vorliegen müssen. Die Implementierung über das *Document Object Model (DOM)* würde die Qualität des Sammlers wesentlich erhöhen und die Vorteile von XML auch an dieser Stelle ausnutzen.

Inhalt der zugehörigen Dateien

Datei ‘rdfHeader.xml’:

```
<?xml version="1.0"?>
<?xml-stylesheet
     type="text/xsl"
     href="http://www.usf.uni-kassel.de/grid/meta/repository.xsl" ?> (1)
<rdf:rdf
     xmlns:rdf   ="http://www.w3.org/rdf/rdf/"                         (2)
     xmlns:dc    ="http://purl.oclc.org/dc/">
```

Datei ‘rdfFooter.xml’:

```
</rdf:rdf>                                                             (3)
```

Die Datei ‘rdfHeader.xml’ enthält die Daten, die zur Einleitung der XML/ RDF-Datei notwendig sind. Neben der Angabe des für die Darstellung der Datei in einem Browser notwendigen Stylesheets (1), sind dies die Deklarationen über die verwendeten Namensräume (rdf: und ds:) (2). Die Datei ‘rdfFooter.xml’ enthält lediglich das abschließende Tag der XML-Datei.

Datei ‘copyRepositoryToWWW.ftp’:

```
# open connection to web server
open 141.51.100.72
# login user
user endejan myPassword
# copy repository to web server
put /grid/meta/repository.xml /home/WWW/grid/meta/repository.xml
bye
```

Die Datei ‘copyRepositoryToWWW.ftp’ ist zuständig für das Kopieren der zusammengestellten Datei an den festgelegten zentralen Ort (hier ein Verzeichnis auf dem Web-Server) und enthält lediglich die hierfür benötigten Informationen über den Server und die zu kopierende Datei.

D.4 Simulationseinstellungen und Datenzugriff

Die folgenden Quelltext-Ausschnitte entstammen dem GLASS-Teilmodell *WaterGAP* und zeigen die Änderungen, die zum Einsatz des Simulationslaufmanagers und des Datenzugriffs-Servers innerhalb dieses Modells notwendig sind.

WaterGAP nutzt für den Kontakt zu den Servern die hierzu entwickelten Klienten-Klassen (SISA_simulationRunMangerClient und SISA_dataAccessClient). Die Deklarationen dieser Klassen sind daher im Quelltext von WaterGAP bekanntzugeben:

```
//----- SISA classes -----
#include "dataAccessClient.h"
#include "simRunManagerClient.h"
//------------------------
```

Im Quelltext von WaterGAP folgen dann allgemeine Einstellungen zu den Servern und zum Klienten (WaterGAP):

```
//----- SISA declarations/definitions -----
char THIS_MODEL[]  = "urn:x-wzusf.sw-mod.WaterGAP-V2,1e.exe-li.hd";  (1)
char simRunName[]  = "urn:x-wzusf.ares-run.SisaTest-V1,0.na.na";     (2)
char daServerIP[]  = "141.51.100.15";                                (3)
int  daServerPort  = 55555;
char srmServerIP[] = "141.51.100.15";                                (4)
int  srmServerPort = 55556;
//----- global simulation-run manager -----
SISA_simRunManagerClient simRunManager;                              (5)

//----- SISA initializations -----
//----- init data-access server (singleton) -----
SISA_dataAccessClient::instance()->setServerInfo(daServerIP,
                                                 daServerPort);      (6)
//----- init simulation-run manager -----
simRunManager.setServerInfo(srmServerIP,
                            srmServerPort);                          (7)
simRunManager.setModelSettingsSpecifications("//Usf-ws14/grid/study/
                 rglass1_3/simulationrun/SimulationParameter.xml");(8)
//--------------------------------
```

Zu den allgemeinen Einstellungen gehören die eindeutige Identifizierung des Simulationsmodells (URN des Modells) (1), die Festlegung auf einen Simulationslauf (der i. d. R. erst zur Laufzeit festgelegt wird) (2) und die Informationen, die notwendig sind, um den Simulationslaufmanager und den Datenzugriffs-Server zu kontaktieren (IP-Adresse und Port) (3) und (4). Von der Klienten-Klasse des Simulationslaufmanagers wird darüber hinaus ein Objekt erzeugt (5), über das die Abfragen der Modelleinstellungen erfolgt (vom Klienten für den Datenzugriff wird nicht explizit ein Objekt angelegt, da die Klasse nach dem ‘Singleton’-Prinzip implementiert wurde[1]).

Nach diesen Definitionen und Deklarationen werden die Server-Informationen gesetzt: für den Datenzugriffsklienten geschieht dies über die Methode *instance()*, die eine Instanz der Klasse *SISA_dataAccessClient* liefert und über

[1]Der Simulationsmodell-Manager ist nach dem Singleton-Muster implementiert, d. h. es gibt maximal *eine* – global zugreifbare – Instanz des Managers innerhalb eines Programms. Näheres zum Singleton-Muster findet sich z. B. in Gamma u. a. (1996).

die anschließend die IP-Adresse und der Port zur Kommunikation mit dem Server gesetzt wird (6).

Beim ersten Aufruf der statischen Klassen-Methode *instance()* erzeugt die Klasse automatisch ein Objekt, speichert eine Referenz auf dieses Objekt in einer statischen Klassen-Variablen und gibt die Referenz anschließend an den Aufrufer zurück. Bei weiteren Aufrufen der Methode wird lediglich die Referenz auf das bereits existierende Objekt zurückgeliefert. Auf diese Weise wird sichergestellt, dass immer nur ein Objekt dieser Klasse in einem Programm existiert.[2]

Die Einstellungen zum Simulationslaufmanager werden ebenfalls über die Methode *setServerInfo()* gemacht (7) – hier allerdings direkt über den Aufruf der Methode für das zuvor manuell erzeugte Objekt. Im obigen Beispiel wird dem Simulationslaufmanager auch direkt innerhalb des WaterGAP-Modells die zu verwendende XML-Datei mit den Modellspezifikationen bekanntgegeben.

Nachdem die grundlegenden Einstellungen gemacht wurden, greift WaterGAP auf die Dienste des Simulationslaufmanagers und des Datenzugriffs-Servers zu:

```
//----- get dataset (old style) -----
//sprintf(filename, "%s/G_CORR_FACTOR.UNF0", options.input_dir);     (9)
//gridIO.readUnfFile(filename, ng, dailyWaterBalance.G_cellCorrFact);

//----- get dataset (SISA style) -----
char datasetURN[255];
// get URN for dataset from simulation-run manager
simRunManager.getSetting(THIS_MODEL, simRunName,                    (10)
                         "dsCorrectionFactor",
                          datasetURN, sizeof(datasetURN));
// get the dataset
SISA_dataAccessClient::instance()->retrieveDataset(datasetURN,      (11)
                                                 "SISA_DS_IMG22UNF0",
                              (char*) dailyWaterBalance.G_cellCorrFact,
                              sizeof(dailyWaterBalance.G_cellCorrFact));
```

Der Quelltext-Ausschnitt zeigt den lesenden Zugriff auf einen Datensatz, wie er in der Originalversion von WaterGAP (9) und wie er unter Verwendung der SISA-Komponenten realisiert ist (10/11). In der ursprünglichen WaterGAP-Version sind der Name der Datei (G_GORR_FACTOR.UNF0) und damit dessen Format (UNF0) fest in den Quellcode integriert. Identifiziert wird der Datensatz allein über die Klassenvariable *options.input_dir*, dessen Inhalt das Verzeichnis festlegt, in dem sich die Datei befindet. Unter Verwendung der SISA-Komponenten wird zum Lesen des Datensatzes zunächst dessen URN vom Simulationsmanager erfragt (10). Dem Simulationslaufmanager muss hierzu der

[2]Der Konstruktor der Klasse ist nicht öffentlich und kann nur indirekt über die Methode *instance()* aufgerufen werden.

Modellname (URN des verwendeten WaterGAP-Modells), der aktuelle Simulationslaufname (ebenfalls ein URN) und der Name der Einstellung (dsCorrectionFactor) übergeben werden sowie eine Variable zur Rückgabe des Einstellungswertes – der Einstellungswert ist wiederum ein URN, der den zu lesenden Datensatz eindeutig identifiziert. Das Lesen des Datensatzes erfolgt dann unter Angabe dieses URNs und des gewünschten Formates (SISA_DS_IMG22UNF0) über den Datenzugriffs-Server bzw. die Klasse *SISA_dataAccessClient* (11).

Der folgende Quelltext zeigt einen Auszug aus der XML-Datei mit den Simulationslauf-Spezifikationen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulationSpecifications SYSTEM "SimulationParameter.dtd"> (1)
<simulationSpecifications>
    <simulationSpecification
      simRunId="urn:x-wzusf.ares-run.SisaTest-V1,0.na.na">          (2)
        <modelSettings simulationModelId=
          "urn:x-wzusf:sw-mod.WaterGAP-V2,1e.exe-li.hd">            (3)
            <simulationStartYear>1961</simulationStartYear>         (4)
            <simulationEndYear>1990</simulationEndYear>
            <simulationTimeStep>1</simulationTimeStep>
            <dsCorrectionFactor>
          urn:x-wzusf:ds-min.CorrectionFactor-V2,1e.unf.hd
        </dsCorrectionFactor>
        <!-- ...>
        </modelSettings>
    </simulationSpecification>
</simulationSpecifications>
```

Zur Validierung der Einstellungen wird unter Punkt (1) zunächst die Datei angegeben, die die *Document Type Definition (DTD)* enthält (s.u.) und über die eine Validierung der Angaben erfolgen kann. Die Spezifikationen für einen Simulationslauf wird von den Marken (Tags) <*simulationSpecification*> und </simulationSpecification> begrenzt. Der Simulationslauf wird über den URN im Attribut *simRunId* identifiziert (2). Zu jedem Simulationslauf werden die geforderten Einstellungen angegeben (3) – die Identifizierung der Modelle geschieht wiederum über die URNs. Das Beispiel zeigt vier Einstellungen für das WaterGAP-Modell, die den Start, das Ende und den Zeitschritt der Simulationszeit sowie den oben bereits angesprochenen Datensatz mit den Korrektur-Faktoren spezifizieren.

Die Definition der Elemente und Attribute, die in der XML-Datei vorkommen dürfen bzw. vorkommen müssen, wird in der DTD vorgenommen:

```
<!ELEMENT simulationSpecifications (simulationSpecification+)>      (1)
<!ELEMENT simulationSpecification (modelSettings+)>                 (2)
<!ATTLIST simulationSpecification simRunId CDATA #REQUIRED>         (3)
```

```
<!ELEMENT modelSettings (simulationStartYear, simulationEndYear,
                    simulationTimeStep, dsCorrectionFactor)>         (4)
<!ATTLIST modelSettings simulationModelId CDATA #REQUIRED>           (5)
<!ELEMENT dsCorrectionFactor (#PCDATA)>                              (6)
<!ELEMENT simulationStartYear (#PCDATA)>
<!ELEMENT simulationEndYear (#PCDATA)>
<!ELEMENT simulationTimeStep (#PCDATA )>
```

Die *simulationSpecifications* bestehen nach Punkt (1) aus mindestens einer *simulationSpecification* – ausgedrückt durch das Zeichen '+'. Zu einer *simulationSpecification* gehört wiederum mindestens ein Satz an *modelSettings* (2), wobei der Simulationslaufname über das Attribut *simRunId* für jeden dieser Einstellungssätze zwingend (#REQUIRED) vorgeschrieben ist (3). Ein Einstellungssatz besteht nach (4) aus genau vier Einstellungen: *simulationStartYear*, *simulationEndYear*, *simulationTimeStep* und *dsCorrectionFactor*. Jede dieser Einstellungen besteht dabei aus einfachen Texteinträgen – gekennzeichnet durch '#PCDATA' (6).

D.5 Simulationssystem

Der folgende Quelltext zeigt einen Ausschnitt aus dem Sicherheitsmodell von GLASS und verdeutlicht das Prinzip der Generierung und Abfrage von Datensätzen.

```
// ----------- Security Model ----------------------------------------

//----- create new security model
GlsSecurityModel *securityModel = new GlsSecurityModel;              (1)

//----- check availability of results
if (securityModel->resultsAvailable(simRunURN)) {                    (2)
  //----- results available
  //----- use results:
  //----- stress
  GlsStressType stress;
  stress = securityModel->getStress(region, year, GlsWater);         (3)
  //----- susceptibility
  GlsSusceptibilityType sus;
  sus = securityModel->getSusceptibility(region, year, GlsWater);
  //----- crisis signal
  GlsCrisisType crisisSignal;
  crisisSignal = securityModel->getCrisisSignal(region, year, GlsWater);
  //----- affected population
  GlsPopulationType pop;
  //----- potentially affected population
```

```
  pop = securityModel->getPotentiallyAffectedPopulation(region, year,
                          GlsWater);
  //----- crisis occurence
  GlsCrisisType crisisOccurence;
  crisisOccurence = securityModel->getOccuredCrisis(region,year,
                                        GlsWater);
  //----- actually affected populaton
  pop = securityModel->getActuallyAffectedPopulation(region, year,
                                                   GlsWater);
  //----- crisis-domain remark
  string remark;
  remark = securityModel->getCrisisDomainRemark(region, year, GlsWater);
  // ...
}
else {
  //----- results not yet available
  //----- generate new dataset
  securityModel->generateDataSet(simRunURN);                        (4)
}
// --------------------------------------------------------------------
```

Unter Punkt (1) wird zunächst eine neue Sicherheitsmodell-Instanz erzeugt. Im Anschluss wird geprüft, ob für einen bestimmten Simulationslauf (*simRunName*) bereits Ergebnisse berechnet wurden (2). Liegen die Ergebnisse bereits vor, können diese direkt über die Operationen zum Datenzugriff (*getStress()*, *getSusceptibility()* etc.) verwendet werden (3). Liegen die Ergebnisse noch nicht vor, wird die Operation zur Generierung der Datensätze aufgerufen (4).[3]

Der folgende Quelltext ist ebenfalls ein Ausschnitt aus dem Sicherheitsmodell und zeigt den Umgang mit dem Simulationsmodell-Manager.

```
//----- get reference to model manager
IamModelManager *modelManager = IamModelManager::instance();         (1)

//----- get references to co-operating simulation models. If required
//----- models are not yet registered, create and register them.

//----- population profile model
_populationProfileModel = (GlsPopulationProfileModel*)
    modelManager->getModelReference("PopulationProfileModel");       (2)
if (!_populationProfileModel) {
  _populationProfileModel = new GlsPopulationProfileModel;           (3)
  modelManager->registerModel("PopulationProfileModel",              (4)
                              _populationProfileModel);
}
```

[3]Die *init()*-Operationen werden automatisch aufgerufen und sind daher im Quelltext nicht aufgeführt.

```
//----- food-stress model                                          (5)
if (_foodSecurityAnalysis) {
  _foodStressModel = (GlsFoodStressModel*)
   modelManager->getModelReference("FoodStressModel");
  if (!_foodStressModel) {
    _foodStressModel = new GlsFoodStressModel;
    modelManager->registerModel("FoodStressModel", _foodStressModel);
  }
}

//----- water-stress model
//...

//----- check, whether model results for water and food stress are
//----- already available for the specified simulation-run. If not,
//----- create data sets.

//----- food-stress model
if (_foodSecurityAnalysis) {
  if (!_foodStressModel->resultsAvailable(                        (6)
                     "urn:x-wzusf.ares-run.SisaTest-V1,0.na.na")) {
    //----- results not yet available -> start simulation run
    _foodStressModel->initGeneralModelService(DataSetGeneration);  (7)
    _foodStressModel->generateDataSet();                           (8)
  }
}

//----- water-stress model:
//...
```

Im ersten Schritt (1) wird eine Referenz auf die (Singleton-)Instanz des Simulationsmodell-Managers beschafft. Anschließend wird vom Modell-Manager eine Referenz auf ein Bevölkerungsprofil-Modell (populationProfileModel) erfragt (2). Sofern innerhalb ein solches Modell noch nicht existiert, wird es erzeugt (3) und beim Modellmanager registriert (4). Die Registrierung erfolgt unter Angabe des Modellnamens und einer Referenz auf das Modell. Wird für die aktuelle Berechnung die Analyse des Nahrungsmittelstresses benötigt, wird die gleiche Befehlssequenz auch für das Nahrungsmittelstressmodell durchgeführt (5) – entsprechendes gilt auch für das Wasserstressmodell. Nachdem die Existenz der Modelle nun sichergestellt ist, wird geprüft, ob die für eine Analyse des Nahrungsmittelsicherheit notwendigen Ergebnisse des Nahrungsmittelstressmodells bereits berechnet wurden (6). Ist dies nicht der Fall, wird eine solche Berechnung initialisiert (7) und gestartet (8).

D.6 Datenzugriff

ArcView-Exportformat für Rasterdaten

Ein für die Datenspeicherung verwendetes Format ist das ASCII-Format, das vom GIS *ArcView* zum Import und Export von Rasterdaten verwendet wird. Die Daten entsprechen den Werten eines regelmäßigen Rasters. Jeder Datensatz beginnt mit einer kurzen Charakterisierung des Rasterdatensatzes. Die folgenden Zeilen entstammen der Beschreibung des Datensatzes mit den Länderkennzahlen, wie sie in IMAGE2.2 verwendet werden.

```
ncols         720
nrows         360
xllcorner     -180
yllcorner     -90
cellsize      0.5
NODATA_value  -9999
```

Ein Rasterdatensatz wird also charakterisiert über die Anzahl der Spalten (ncols – number of columns) und Zeilen (nrows – number of rows), die Angabe des x- und y-Wertes der linken oberen Ecke der geographischen Abdeckung (x/yllcorner – x/y lower left corner) und die Angabe der Größe einer Zelle (cellsize; hier 0.5 Grad). Darüber hinaus kann optional der 'Wert' angegebenen werden, der für unbekannte Werte innerhalb des Datensatzes verwendet wird (NODATA_value). Der NODATA-Wert wird im Datensatz der Länderkennzahlen beispielsweise für die Zellen der Ozeane eingesetzt (die in IMAGE2.2 keinem Land zugeordnet sind).

Nach der Beschreibung des Datensatzes folgen die eigentlichen Daten, indem zeilenweise die Werte jeder Zelle des Rasters von 'oben links' (-180°/+90°)nach 'unten rechts' (180°/-90°) angegeben werden.

D.7 Geodatenverarbeitung

Der folgende Quelltext zeigt die Verwendung einiger Operationen der Klasse zur Geodatenverarbeitung innerhalb einer Version des Wasserstressmodells.

```
void GlsWaterStressModel::calculateStressAreaBased(void)
// calcutate water stress based on current area below threshold
{
  //----- calculate relative current value [% of mean]
  *_currentRelative_themePtr =  *_currentValue_themePtr             (1)
                             / *_runningMean_themePtr * 100;

  //----- calculate stress [% of region area where deviation>threshold]
```

```
  //----- get cells with values below threshold                         (2)
  *_cellsBelowThreshold_themePtr = (*_currentRelative_themePtr <
                                     _availabilityThreshold);

  //----- calculate region area for all cells with values<threshold
  *_affectedCellArea_themePtr =  *_cellsBelowThreshold_themePtr
                               * *_cellArea_themePtr;

  _affectedCellArea_themePtr->zonalSum(*_region_themePtr,                (3)
                                        *_affectedRegionArea_themePtr);

  //----- calculate stress [area fraction in % of total region area]
  *_affectedRegionAreaFraction_themePtr =  *_affectedRegionArea_themePtr
                                          / *_regionArea_themePtr
                                          * 100;

  *_regionStress_themePtr = * _affectedRegionAreaFraction_themePtr;
}
```

Die dargestellte Methode (calculateStressAreaBased) berechnet den flächenbezogenen Wasserstress. Alle Geodatensätze werden hier über Objekte einer Klasse ‘theme’ repräsentiert (die Definition der Objekte findet an anderer Stelle statt). Die Geodaten-Objekte im Wasserstressmodell repräsentieren jeweils 66896 Rasterzellen (IMAGE2.2-Format). Zur Berechnung des Stresses wird zunächst die relative Wasserverfügbarkeit (currentRelative) für jede Rasterzelle berechnet (1). Hierzu wird eine Division mit den Geodaten-Objekten durchgeführt, um das Ergebnis anschließend mit einem skalaren Wert (100) zu multiplizieren. Die Berechnungen werden dabei automatisch für jede der 66896 Rasterzellen durchgeführt. Anschließend folgt eine logische Operation mit zwei Geodaten-Objekten (2). Nach einer weiteren Multiplikation erfolgt die Bildung der zonalen Summe (3).